PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 12/14

A1

(11) International Publication Number: WO 90/13084

(43) International Publication Date: 1 November 1990 (01.11.90)

(21) International Application Number: PCT/US90/02113

(22) International Filing Date: 18 April 1990 (18.04.90)

(30) Priority data:
340,886 19 April 1989 (19.04.89) US

(71) Applicant: EMPIRICAL RESEARCH SYSTEMS, INC. [US/US]; P.O. Box 583, Tacoma, WA 98401 (US).

(72) Inventor: JONES, Richard, P. ; 7930 East Side, N.E., Tacoma, WA 98422 (US).

(74) Agents: GARRISON, David, L. et al.; 2100 Fifth Avenue Building, Seattle, WA 98121 (US).

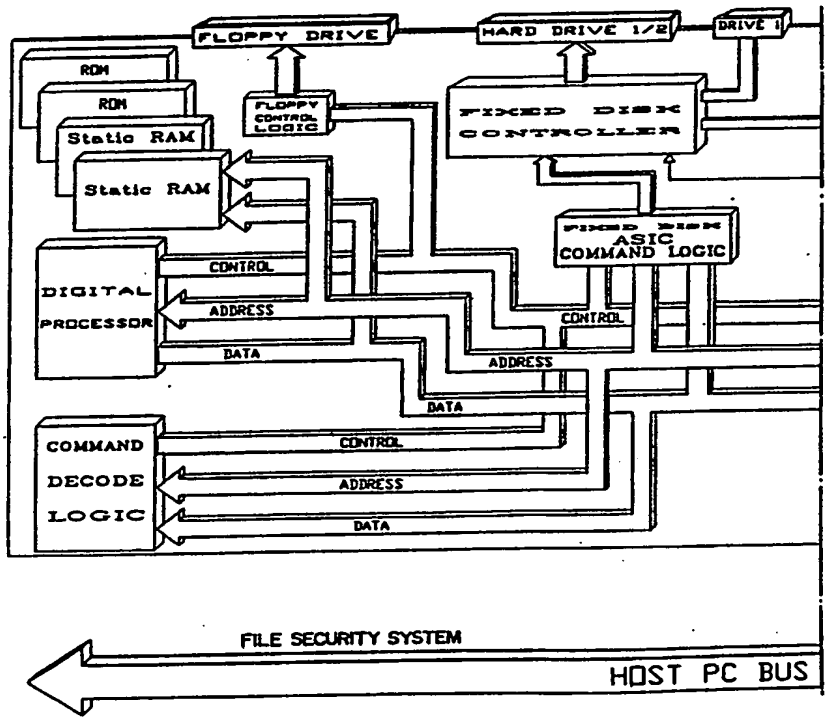
(81) Designated States: AT (European patent), AU, BE (European patent), CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB (European patent), IT (European patent), JP, KP, KR, LU (European patent), NL (European patent), SE (European patent), SU.

Published
With international search report.

(54) Title: COMPUTER FILE PROTECTION SYSTEM

(57) Abstract

The invention is a system for protecting the security of computer files. It has hardware elements, including a programmable auxiliary memory and control unit along with associated software elements. The security subsystem is installed on the host computer bus so that it resides in the control logic, address, and data signal path between the computer storage device and central processing unit. The security system is accessible by the computer operating system only during installation and initialization. Thereafter it is inaccessible to or by the operating system. Supervisor determined criteria for access permission to read, write and execute files are entered into the auxiliary memory system where they are protected from alteration. The security system will deny access to users with invalid entry criteria and refuse to write data to the file storage device when unauthorized operations have been performed. When breaches of these types occur the security system can lock the computer against further activity until it is released by entry of a master password from supervisory or security personnel. The system maintains a protected area in the computer memory device where, among other data, file signatures of all valid files are retained. The protected area of memory also maintains appropriate signatures of all internal files in the security system so that they can be automatically checked for integrity.



+ See back of page

DESIGNATIONS OF "DE"

Until further notice, any designation of "DE" in any international application whose international filing date is prior to October 3, 1990, shall have effect in the territory of the Federal Republic of Germany with the exception of the territory of the former German Democratic Republic.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	ES	Spain	MC	Monaco
AU	Australia	FI	Finland	MG	Madagascar
BB	Barbados	FR	France	ML	Mali
BE	Belgium	GA	Gabon	MR	Mauritania
BF	Burkina Fasso	GB	United Kingdom	MW	Malawi
BG	Bulgaria	GR	Greece	NL	Netherlands
BJ	Benin	HU	Hungary	NO	Norway
BR	Brazil	IT	Italy	RO	Romania
CA	Canada	JP	Japan	SD	Sudan
CF	Central African Republic	KP	Democratic People's Republic of Korea	SE	Sweden
CG	Congo			SN	Senegal
CH	Switzerland	KR	Republic of Korea	SU	Soviet Union
CM	Cameroon	LI	Liechtenstein	TD	Chad
DE	Germany, Federal Republic of	LK	Sri Lanka	TG	Togo
DK	Denmark	LU	Luxembourg	US	United States of America

COMPUTER FILE PROTECTION SYSTEM

BACKGROUND OF THE INVENTION

The present invention is a method for protection of computer files from unauthorized access and/or modification and from unintentional damage. It is particularly useful for protection of files against malevolent tampering and sabotage.

A problem of serious and potentially disastrous proportions exists in the protection of computer files from unauthorized modification. This ranges from unauthorized but benign entry by unethical computer buffs, who regard it as a personal challenge to find ways to enter a system, to deliberate and criminal sabotage of stored data and software. The extent of computer crime has grown markedly as criminal elements, now aware of the possibility for ill gotten gain or vengeance, and have achieved a hitherto unknown level of sophistication. Much of the computer crime that does occur is little publicized. This is to minimize its consequences and to avoid encouragement of others who might find it attractive. In addition to the possible enormous consequences for business, education, and general government operations, the implications for defense could be of the nature of a major national disaster.

An especially pernicious form of file modification is known as a "virus". The analog to a biological virus is readily apparent. A computer virus is designed to attach itself to a program already on the computer. The result is a program that is "infected". This usually occurs in a manner that, at least initially, is unapparent to the legitimate user. The infected target files are usually unchanged until some predefined event or events take place. At this time the virus embedded in the infected files activates. The action taken when the virus is triggered may range from a harmless message flashed throughout the system to the complete destruction of all files in the infected system. A virus must, by definition, modify a file stored in the system in order to propagate itself. A virus recently propagated within three days throughout a nationwide computer network and caused damage in excess of $10 million.

Other forms of computer file vandalism are known within the computer profession as "worms", Trojan hors s", and "bombs". All such programs modify the file system in some manner in order to perform their intended function. While the above terms have specific meaning to computer scientists, for the sake of convenience they will all be classified here as "viruses" since their operation, intent and methods of prevention are in most ways very similar.

Computer files are also subject to innocent errors resulting from accidental and unintended mistakes. Within a given environment the effect may be as damaging as a virus. However, such errors generally do not spread to other computers. A particularly vulnerable environment is one in which software development is in progress. Generally there is no mechanism for protecting files on a computer system from damage by errant programs. Valuable files can easily be destroyed requiring many hours, days or weeks for reconstruction.

Another source of innocent error is accidental erasure or modification of files. This can result from a simple mistake on the part of the operator and may or may not be salvageable. Most of the commonly used operating systems for individual or networked computers allow files to be erased or modified with simple commands that do not prompt or question the user before proceeding.

Computer security has itself become a recognized specialty within the profession. The most common method of minimizing problems from intrusive sabotage is to incorporate software in the computer system that checks for known types of viruses and/or periodically checks the integrity of the files in the system. There are a number of variations on the software approach to protection. One method of checking file integrity is to perform a test of each file which results in a unique "signature" for the file. This method is reasonably robust but it is somewhat time consuming. Most often, the signature is generated using a Cyclic Redundancy Code (CRC) algorithm. This test does nothing to cure a file which might have become infected but it does identify files which have been infected since they were last tested. To be truly effective the test should be run each time a file is accessed. However, in most cases this would impose such a large overhead as to make the system non-productive.

A second method is to incorporate a software program which checks each file as it is used for a set of known types of infection. Several problems exist with this approach. First, a number of viruses are self-modifying. By that is meant that they change their characteristics specifically in order to thwart this kind of protection. Second, new viruses unknown to the protection program may be introduced into the file system and these will not be recognized. A third problem is that of overhead. It may be so great as to significantly reduce the usefulness of the system.

Another method provides a hardware module which can be programmed to write protect the entire file system. This method is clearly foolproof but poses such cumbersome limitations that it has only limited usefulness. Most business or scientific applications and virtually all program development environments require the ability to modify files. As one example, the files in a database application are usually continually updated by new additions and deletions. The hardware write protect approach appears to have merit only in those unusual instances where an application does not require the file system to be modified. If software support is used to determine when a file can be modified, it is susceptible to the same problems and disadvantages of the other software approaches.

A fourth method uses passwords and other user specific security protection to limit access to the file system. This is desirable and should be common practice in most computer networks. But it does not prevent the problem of virus entry. One of the most destructive viruses reported to date infected over 5000 computers, all of which had a password and user permission-based file system. The most common use of this type of protection is found on computers based on the UNIX operating system. Unix is a trademark of AT&T Information Systems, New York, New York, for a linked multi-workstation computer system. In regard to accidental file erasure, a number of products are available with an "unerase" feature. These take advantage of the way most files are removed from a directory by the operating system. The operating system simply indicates that the storage space of the deleted file is now available for n w files, without actually physically erasing the earlier material. The unerase software restores the deleted file name back into the dir ctory. However, it can function successfully only if the storage space occupied by the deleted file has not been overwritten.

The process of attempting to recover an accidentally erased file is time consuming and can sometimes result in a corrupt d file even under the best of circumstances.

Finally, any software based system of virus protection has an inherent flaw that can itself be fatal. The very software that is intended to protect against infection can itself be the source of a virus. This very problem recently occurred with a suite of commercial programs, touted as the ultimate in anti-virus protection. The case in point was apparent sabotage by a disgruntled employee of the software firm marketing the protection system. An untold number of infections occurred and the manufacturer now faces an enormous liability for damage caused by his product.

Any security system, intended to provide protection for file systems, which is accessible to the general user through standard system resources can potentially be breached. Whether software based or hardware based, if the protection system can be accessed via normal system resources, then it can be bypassed or, even worse, used to camouflage a virus. A user who thinks the file system is protected is often complacent and less alert to the possibility of an infection. This often leads to a virus doing extensive damage before it is even noticed.

Cognizant of the above noted shortcomings in existing file security systems, the present invention represents a major improvement that greatly reduces and tightly controls the number of potential access points for virus entry without compromising convenience and utility for the general user.

SUMMARY OF THE INVENTION

The present computer file security system has both hardware and software elements. Unlike any other system known to the present inventor, once installed, the protective elements of the system are completely inaccessible to the general user. The system provides essentially absolute protection against inappropriate modification of all designated files held within the computer memory device.

The file system protection process operates by intercepting the file system data path between the central processing unit and the file storage or memory device. The requested operation is processed according to the criteria established by the supervisory and/or security personnel of the computer system. An elaborate and virtually unbreak-

able system of access eliminates any chanc of file corruption by a general user.

The security subsystem is accessible by the computer operating system for initialization and modification only during an installation stage. After that time the security subsystem is inaccessible to or by the operating system.

Each time the security subsystem sends an error message to a user it is checked for file integrity. At this time it takes possession of and disables all other access to the computer central processing unit.

In general, the hardware elements of the system are integrated with a controller for mass storage of the file system, although this is not always necessary. The process can be just as easily incorporated into a local network (LAN) controller, a communications controller, or a main processor board for a system. In its broadest form, the present file security system could be applied to a wide variety of situations where access to critical data must be controlled.

The invention includes a programmable auxiliary memory and auxiliary control unit. These can be attached to the host computer bus in a manner so that they are in the control logic, address, and data signal path between the central processing unit and the file storage system. However, once installed in the computer system, the file security system is inaccessible to or by the host computer operating system. Access to the file security system is possible only by using a unique password held by the appropriate supervisory and/or security personnel. Access may be established on a hierarchical basis so that for some designated operations more than one individual must enter passwords in proper sequence.

The supervisory personnel will choose and enter the appropriate criteria for access permission to read, write, and execute operations for all files to be protected. These criteria will be specific to each user or user group. The file security system can be programmed for graduated levels of security and lockout for various types of users.

Upon receiving valid user identification, the auxiliary memory and control unit will indicate to the host computer operating system which files are accessible to that user and the nature of the operations that can be performed on the files. Similarly, users with invalid entry criteria for the files requested will be denied entry and the file

security system will refuse to allow data to be written into the host computer file system when unauthorized operations have been performed.

In many linked computer systems each computer central processing unit has its own associated file system. Usually, the file systems of every individual computer in a linked system are electronically available to every other computer in the system. Most preferably, the file security system of the present invention should be used to protect each file system in a given linked computer system. This would require associating a security system with each file storage device in the system. However, it is quite possible to protect some of the computers in the system while leaving others unprotected. Some linked systems are constructed with a central file storage device, or file server, which is tied to a number of different computers, each having its own central processing unit. In this case a single file security system is adequate to protect the entire network. The file security system of the present invention is equally suitable for use on a single terminal computer.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a simplified block diagram of a hardware card showing one implementation of the present invention.

FIG. 2 is an index to the symbols used in the following process flow diagrams.

FIGS. 3-6 are process flow diagrams showing the basic logic of the file security system.

FIGS. 7-18 are subprocesses associated with the basic process logic.

FIG. 19 shows a subprocess used within the various other subprocesses.

FIGS. 20 and 21 show direct memory access read and write subprocesses used within the various other subprocesses.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Most of the terms and abbreviations used herein are in common use and well understood by those skilled in computer sci nce. However, the following definitions will unequivocally set their context for the present invention.

ADR, ADDR	address
ATRIB	attribute
AUTH	authorized
BIOS	basic input/output system
BLD	build
BPB	Bios parameter block
BUS	bus (command, address, or data)
BZY	busy
CHG	change
CHK	check
CMD	command
CNT	count
CNTRL	control
CNTX	number of increments successfully transferred
CPU	central processing unit
CRC	cyclic redundancy check
CRTL	control
DAT	data
DEV	device
DMA	direct memory access
DPAC	data packet
DSTR	32 bit starting logical sector
ERR	error
ERRS	errors
FLG	flag
FRE	free
FSS	file security system
GEN	generic
GET	get
HD	head
HDR	header
IBM-PC	a personal computer manufactured by International Business Machines Corporation.

ID	identity or identification
ILL	illegal
INIT	initialize
IOCTL	input/output control
LBL	label
LOG	logical
MAJ	major
MEM	memory
MS-DOS	Microsoft Disk Operating System
MTY	empty
NUM	number
PAC	packet
PARM	parameter
PC-DOS	IBM Personal Computer Disk Operating System
PHYS	physical
PREV	previous
PTR	pointer
QUE	queue
RD	read
RH	request header (from MS-DOS)
RMV	remove, removable
SECT	sector
SEEK	seek
SET	set
STAT	status
STR	starting
SYS	system
TBL	table
TRK	track
TYP	type
TSR	terminate and stay resident
USR	user
VFY	verify
VOL	volume
WR	write
XADR	transfer address
XCNT	number of increments requested to be transferred
XFR	transfer

XSS starting section for transfer

The term "bus" or "host computer bus" refers to the electronic paths within the host computer that carry address, control, and data signals. The "address bus" is a collection of electronically continuous lines used to provide a unique location for access to a system resource such as memory or input/output devices. The "data bus" is a similar collection of lines used to pass information between locations determined by the address bus. The "control bus" is a similar collection of electronically continuous lines used to define the kind of operation to be performed on a system resource. As an example, the "memory read line" (MEMR) of the IBM-PC control bus specifies that the operation to be performed is to read the contents of memory at the location specified by the address bus and place that information on the data bus. While a bus is most usually regarded in terms of electrical conductors, it should be considered more broadly in terms of the present invention. As used herein the term "bus" should be considered to also include any alternate means of data interface with the CPU of the computer system that would serve the same purpose as conventional electrical conductors. A fiber optic system could be one such alternative. In its broadest context a "bus" is any means or method that carries information between the computer system and any peripheral devices and which provides control and data to a file storage device. It does not matter whether the device is internal or external to the computer itself.

A "device" is any physical piece of equipment integral with the computer system, such as a mass storage unit, printer, console, etc. In terms of internal communication within the computer a "device" is treated or considered in the same manner as a file would be. A "logical device" is a method of mapping a physical device to hide its real characteristics from the operating system. Although multiple physical devices could be mapped to a single logical device, a more common approach is to map a single physical device to multiple logical devices; e.g., a single 64 megabyte hard disk can be mapped to two 32 megabyte logical devices.

"Direct memory access" (DMA) is used to move data between memory and a device by taking control of the address bus, data bus and control bus. The CPU is disabled during the DMA cycle.

A "logical sector address" enables the use of logical addressing by operating systems to access mass storage devices and hides the

physical characteristics of the device. Logical addressing provid s significant improvements in device indep ndence of the operating system. The conversion of logical address to physical address is device dependent.

"Parameters" when used with a process are data passed to the process. When referring to a device they are the constants that define the device; e.g., the number of heads, sectors per track, etc.

A "pointer" is an "object" or variable used to hold the address of another object; i.e., it "points" to the other object. The context of the term "object" is a variable used to hold an address within the addressable range of values of the main CPU. The pointer is used to indirectly store or retrieve other variables.

"Programmed I/O" describes input or output operations which are controlled by the CPU. This type of I/O is generally used to write to the device control register and read from the device status register. Data transfers are usually handled by direct memory access.

A "queue" is a form of temporary storage used to allow asynchronous data flow between the source and destination. A common form of queue is to allow data to continue to be placed in a waiting list, to avoid having the data source wait, while a slower process operates on the list. The queue is monitored to prevent overflow. If the queue fills the process sourcing the data must wait until the data consumer has removed data.

A "register" is a location used to hold information associated with an operation. "Device registers" hold either control information, status of device after operation, or data. A "control register" is a device register used to select the function to be performed by the device. A "data register" is used to hold data for transfer to and from the data bus under program control.

A "terminate and stay resident" (TSR) program is one that remains in memory after initial activation. The file security system uses such a program to communicate with the user; e.g., "Access Not Authorized" or "Invalid Password".

Before describing the present invention in detail, it could be helpful to the general reader to very briefly review the essential elements of a digital computer most closely related to operation of the invention. All computers have a central processing unit (CPU) and a file storage device. The latter may include a fixed or "hard" disk, one

or more flexible or "floppy" disks, a magnetic tape unit, or an optical device-such as a laser read compact disk unit. The CPU and storage device are joined electronically by a bus system that carries address, control and data signals. The electrical path may not always be electrically direct; i.e. there may be intervening operations on the signals, but the bus maintains the main route of electronic communication between the two units. Other devices such as disk controllers, etc. are essential to operation but are peripheral to the present explanation.

In addition to the basic electronic "hardware", the computer must have a software package known as an "operating system". This serves to enable and supervise the flow of signals between the various hardware elements of the computer, such as the CPU and file storage device, and between the computer and operator. The operating system is not an operations software program, such as a data management or spreadsheet tool would be, but it is essential to their use.

A number of well known operating systems are available for computers of different types and capabilities. Two of the most popular products are very similar and are intended for use with personal computers. These are known as MS-DOS and PC-DOS (Microsoft Operating System and Personal Computer Operating System). MS-DOS is a registered trademark of Microsoft Corporation, Redmond, Washington and PC-DOS and IBM are registered trademarks of International Business Machines Corporation, Armonk, New York. The present invention is suitable for use with these as well as other operating systems such as CP/M, VMS, or UNIX. CP/M is a registered trademark of Digital Research, Pacific Grove, California. VMS is a registered trademark of Digital Equipment Corporation, Maynard, Massachusetts. UNIX is a trademark of AT&T Information Systems, New York, New York. This list should be considered as exemplary and is not inclusive of the many other operating systems suitable for use in conjunction with the present invention.

Whatever the operating system, there is a minimum set of file system operations that must be available. This set must include:

(1) A means of listing the files available on the system to a terminal or display;

(2) A method of providing unique names and internal addresses for the files;

(3) The ability to create a new file, to write information to a file, to read information from a file, and to change the size of a file by adding or deleting information; and

(4) The ability to remove a file from the system.

To implement this minimum set of file system operations there is an attendant set of hardware and software functions. While these functions vary in complexity and capability the following is a representative minimum set.

(1) A means of storing files. Usually this is a mass storage device such as a fixed disk or one of the other types previously noted. The file system must be capable of handling files in a manner consistent with the requirements of the operating system.

(2) A method of formatting the storage medium to meet the needs of the file system. The format generally involves sectioning the storage medium in such a way that the translation between a logical location and a physical location is minimized. The operating system deals with logical addresses of information while the actual storage device operates on physical addresses.

(3) A means of passing commands to the storage subsystem. This typically is a hardware card that interfaces the addresses and data from the system bus to the storage subsystem hardware.

(4) A means of implementing commands to (a.) position or index the storage media to a known starting position, (b.) read from a specific location on the medium and make the information available to the system bus, and (c.) write information from the system bus to a specific location on the medium. In the latter two cases the information is moved directly into and from the system read/write memory by a mechanism known as direct memory access (DMA).

The file security subsystem may be likened to a gate and gate tender on the pathway linking the CPU and file storage subsystem. Only information that meets a set of predefined criteria is allowed to pass. Once placed in position, the gate is impregnable to any changes in the criteria that an unauthorized person might attempt to be made via the operating system. Changes can only be made by an appropriate security director having the master access password.

This location in a computer is unique for a file security system. The only other subsystem placed astride the main bus in similar fashion is an encryption/unencryption device. It is emphasized here

that the file security subsystem in not, nor is it in any way analogous, to an encryption device. It may include an encryption device but this would be entirely ancillary to its main function and operation.

Because they are so well known and in such common use, the description of the invention that is to follow will be based on the use of the PC-DOS or MS-DOS operating systems used with IBM or IBM-type personal computer equipment. However, it should be understood that this is done for the sake of convenience and simplicity of description and the invention should not be considered as limited to these or any other operating systems or computer equipment.

The file security subsystem has a cache memory system attached to the host computer main bus. This provides the elasticity needed to minimize delays associated with the parsing of information by the protection process. While a specific bus transaction is being evaluated, additional transactions are queued up for subsequent processing.

During installation of the file security subsystem (FSS), a set of file access criteria are entered and stored in nonvolatile memory in the FSS and also written to a portion of the host computer file storage device which is subsequently marked as inaccessible to the operating system. These criteria are used by the protection process to determine the type of access authorized on a specific system. After installation the file security system is accessible only by use of a master password that will presumably be known only by an appropriate security dir ctor or system administrator.

The file access criteria will include the names of files which are to be protected at all times. These will generally be the basic executable files that constitute the application for which the system is intended, as well as any utility and system files used by or in support of the application. The access criteria will also contain the names of files that are allowed to modify specified files along with the type name of the specific files which may be modified.

Other access criteria are relatively conventional. The user may be required to enter a login code which can be associated with a specific directory, group of files, or both. The login code can then be used as a test for a password; i.e., if the password given is not authorized for the login group entered, the user will be denied access, even if the password is otherwise valid. This, in effect, provides a double password system.

Operation of the File Security System

During startup, the file security system will check the files associated with the operating system for consistency. This is done by comparing the file signatures of the active files with those held in an archival status in a portion of memory within the file storage device that is inaccessible to the operating system. The same check can be made for any change in file signature of all executable files. As was noted earlier, a unique signature for each file can be generated; e.g., by using a cyclic redundancy code algorithm. If any inconsistency is found during startup, the file system storage device is write protected by the file security system and the user notified. The file which caused the warning is identified and the system is effectively locked until corrective action is taken. This might include removal and replacement of the affected file or an override by the system administrator who must use the master password. If the decision is made to override, the file signature can be updated so that the next startup will accept the modified file.

Programs run by the user are consistency checked as they are loaded into the system memory for execution. As in the startup phase above, any corruption of a file will result in disabling the write circuitry, user notification, and system lockup pending supervisory action.

The file security system will detect any attempt to install new files on the system via the disc drives, serial interface, supported local area networks, or by any executable file. This applies even if the new file is generated on the workstation itself.

The system administrator has a great deal of flexibility in setting security levels. Certain features can be disabled or expanded at the discretion of the administrator.

An optional feature of the file protection system is creation of a transaction log recorded in the protected portion of the file storage device. This can include such items as attempted entries to the system, entries attempted using bad login codes or passwords, terminal locations, time and date, etc. The transaction log is accessible only to supervisory people possessing the master password to the file security system.

It can be seen from the above description that by placing the file security system in the data path between the CPU and storage device, and by making it invisibl to the operating system, the computer file system is protected against deliberate tampering from either local sources or those at other linked locations. Corrupted files are refused entry by write protecting the storage device. Even in the event that such files should somehow find entry, they are detected and identified and the system is locked before the corrupted files can do the intended damage. This protection is equally important in guarding the system from damage by benign errors that frequently occur during program development.

A very important part of the file security system is its terminate and stay resident (TSR) program. In the event of entry being denied to the system for some reason, or an unauthorized operation being performed, the TSR program will send an error message to the user. However, before each use the TSR program is itself checked for file integrity. During the time the TSR program is active it takes possession of and disables all other access to the CPU. Further the TSR program directly accesses the keyboard, bypassing all system software.

Specific operation of the file security system hardware and software can best be understood by now referring to the Figures. FIG. 1 is a simplified block diagram showing one version of a hardware implementation. Given this diagram, the specific construction of the unit will be evident to one skilled in computer science.

FIG. 2 is a symbol legend to the process logic diagrams shown in FIGS. 3-21. Reference to these diagrams will convey a full best present mode disclosure to one skilled in the art. A step-by-step verbal description is not only not necessary but would be redundant.

The basic outline of the process logic is shown in FIGS. 3-6. FIG. 6 ties by the various offpage connector symbols to subprocesses shown in FIGS. 7-18. FIG. 19 is a subprocess used within the various other subprocesses. FIGS. 20 and 21 show direct memory access subprocesses also used in the various other subprocesses.

Having thus disclosed the best mode known by the inventor of making and using his invention, it will be evident to those skilled in the art that many variations are possible without departing from the spirit of the invention. The invention should be considered as being limited only as it is described in the appended claims.

CLAIMS:

1. In a computer file protection method for a host digital computer, said computer having a file storage device and a central processing unit connected electronically by a bus carrying control logic, address, and data signals, said computer further being supplied with operating system software, the improvement which comprises:

providing a file security subsystem for said digital computer, said security subsystem further comprising a programmable auxiliary memory and control unit attachable to the host computer bus in a manner so that it resides in said control logic, address, and data signal path between said storage device and central processing unit, said security subsystem being accessible by the computer operating system for initialization and modification only during an installation stage of the security subsystem but following said installation stage, during computer system operation, the security subsystem is inaccessible to or by the operating system,

the auxiliary memory system being adapted for receiving and retaining supervisor entered criteria for access permission for read, write and execute operations for all files to be protected,

so that upon receiving valid user identification the auxiliary memory and control unit will indicate to the host computer operating system which files are accessible to that user and what operations may be performed upon said files, said auxiliary control unit denying entry to users with invalid entry criteria and refusing to write data to the file storage device when unauthorized operations have been performed.

2. The computer file protection method of claim 1 which further includes the internal capability of self checking its associated file integrity.

3. A computer file protection method which comprises:

providing a host digital computer, said computer having a file storage device and a central processing unit connected electronically by a bus carrying control logic, address and data signals;

supplying operating system software for said computer;

further providing a file security subsystem for said digital computer, said security subsystem further comprising a programmable auxiliary memory and control unit attachable to the host computer bus in a manner so that it resides in said control logic, address, and data signal path between said storage device and central processing unit, said security subsystem being accessible by the computer operating system for initialization and modification only during an installation stage of the security subsystem but following said installation stage, during computer system operation, the security subsystem is inaccessible to or by the operating system,

the auxiliary memory system being adapted for receiving and retaining supervisor entered criteria for access permission for read, write and execute operations for all files to be protected,

so that upon receiving valid user identification the auxiliary memory and control unit will indicate to the host computer operating system which files are accessible to that user and what operations may be performed upon said files, said auxiliary control unit denying ntry to users with invalid entry criteria and refusing to write data to the file storage device when unauthorized operations have been performed.

4. The file protection method of claim 3 wherein the file security subsystem after installation is accessible from the host computer only by entry of a proper master password.

5. The file protection method of claim 3 wherein the file security subsystem during installation creates a protected area within the file storage device inaccessible to the operating system but accessible to the security subsystem.

6. The file protection method of claim 5 including determining a unique file signature for all pertinent files within the system, said signatures being retained for archival reference in the protected storage area, comparing the archival signatures with the current signatures of user requested files prior to permitting user access to the files, and write protecting the storage device if file signatures do not correspond.

7. The file protection method of claim 5 including checking the file signature of files load d into the central processing unit for execution, comparing said file signatures with archival signatures held within the protected storage area, and write protecting the file storage device if file signatures do not correspond.

8. The file protection method of claim 6 which further includes locking the computer system from further activity when said file signatures do not correspond, said computer system remaining disabled until unlocked by a person with access to a master password.

9. The file protection method of claim 7 which further includes locking the computer system from further activity when said file signatures do not correspond, said computer system remaining disabled until unlocked by a person with access to a master password.

10. The file protection method of claim 3 which further includes taking possession of the host computer central processing unit by the file security subsystem and disabling all other access to said central processing unit at such time as the security system detects invalid entry criteria or an unauthorized operation.

11. The file protection method of claim 3 which further includes checking file signatures of the internally stored parameters in the file security system for integrity prior to notifying a user that entry has been denied or an unauthorized operation has been attempted.

12. The file protection method of claim 3 which further includes checking the file access criteria for consistency from archival file signatures held within the protected storage area.

13. The file protection method of claim 3 which further includes creating a transaction log in the protected storage area, said transaction log being accessible only to a person having a master password.

14. The file protection method of claim 3 in which said said supervisor entered criteria are specific for each user or user group.

FIG. 1

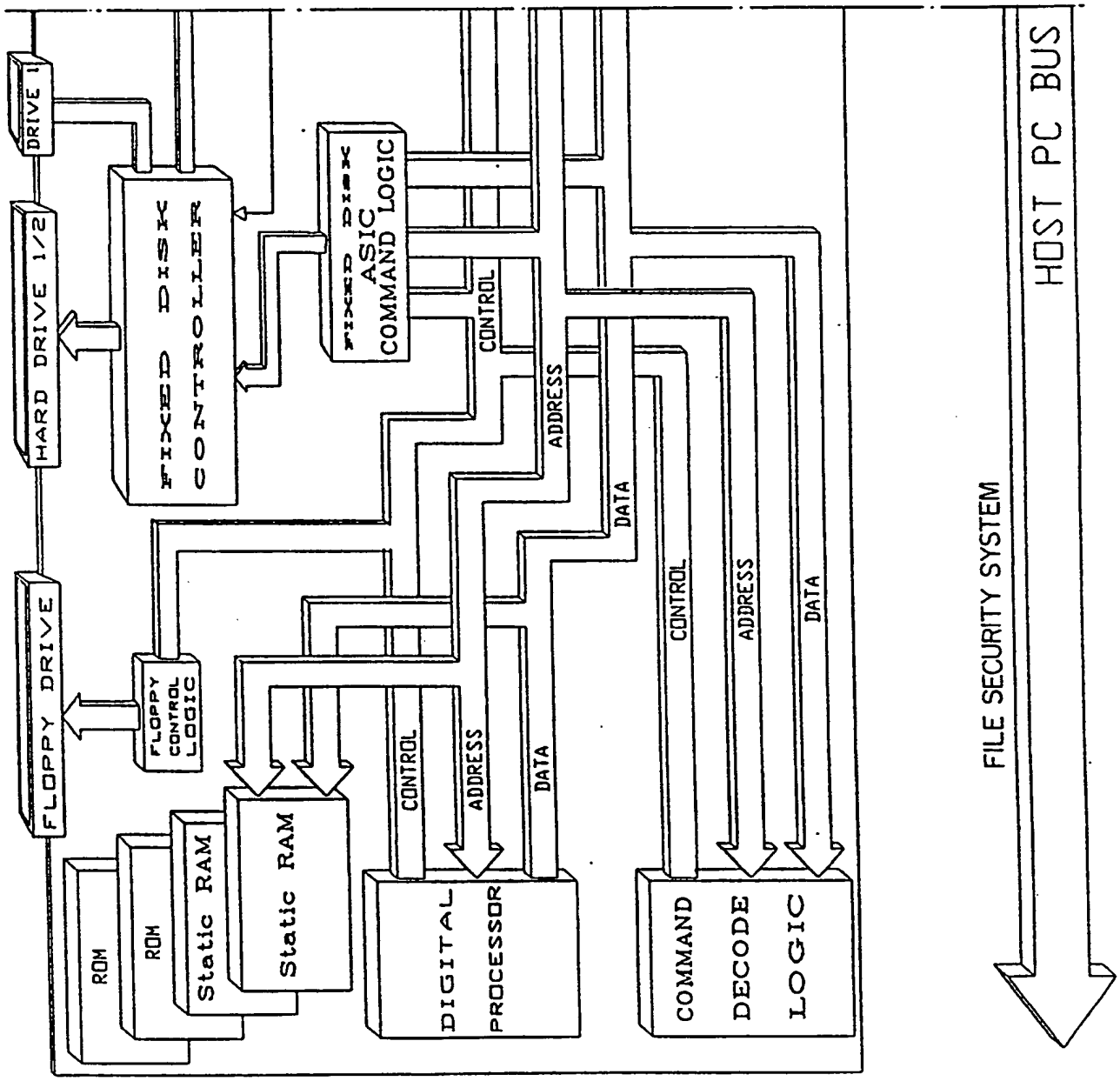
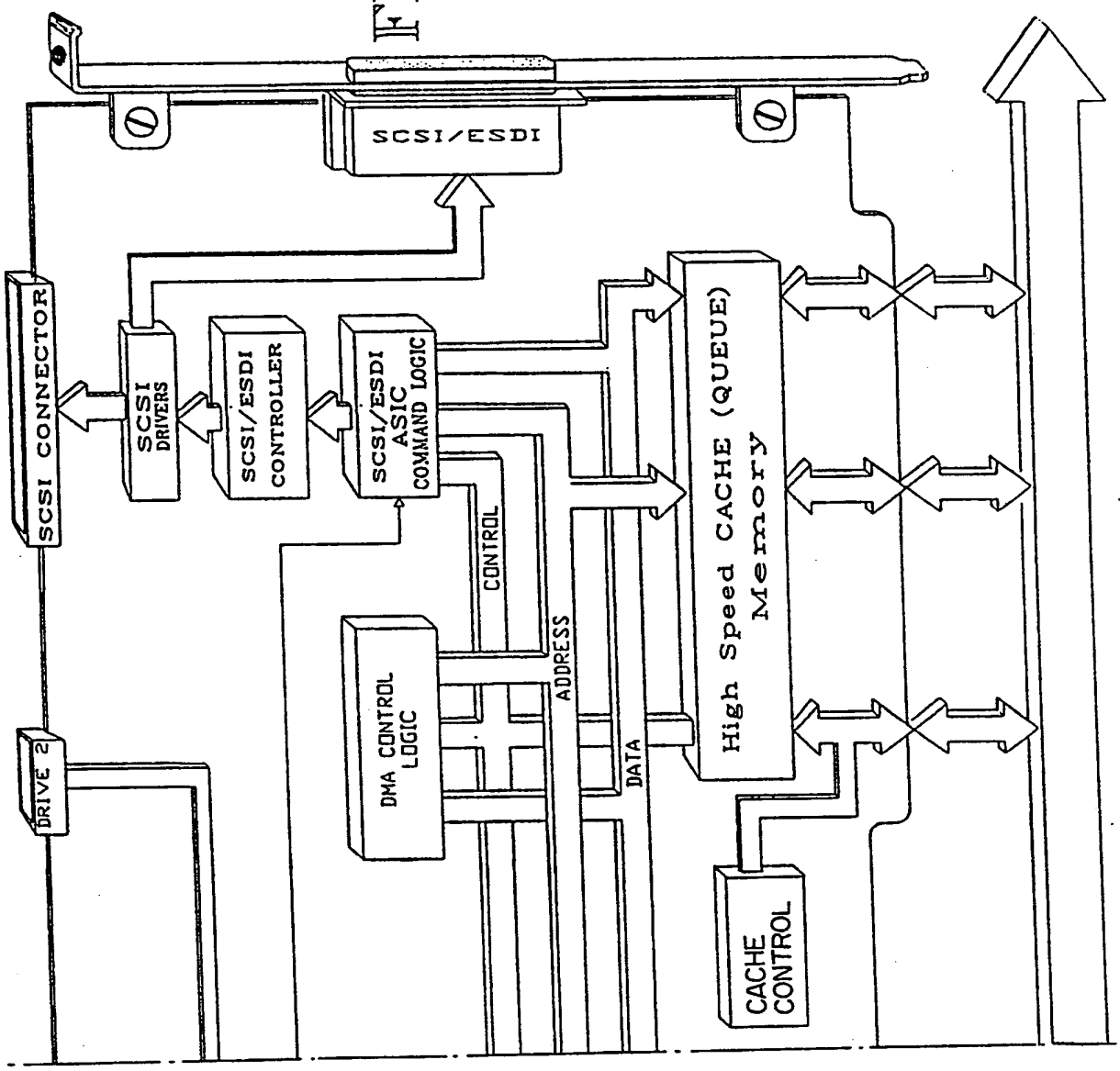

HOST PC BUS
FILE SECURITY SYSTEM
DRIVE 1
HARD DRIVE 1/2
FLOPPY DRIVE
FIXED DISK CONTROLLER
FIXED DISK ASIC COMMAND LOGIC
FLOPPY CONTROL LOGIC
ROM
ROM
Static RAM
Static RAM
DIGITAL PROCESSOR
COMMAND DECODE LOGIC
CONTROL
ADDRESS
DATA
CONTROL
ADDRESS
DATA
CONTROL
ADDRESS
DATA

FIG. 1 CONT.



SUBSTITUTE SHEET

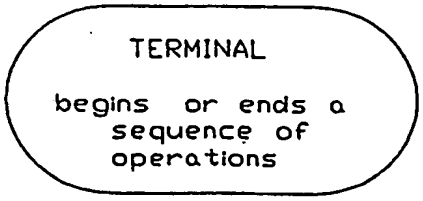
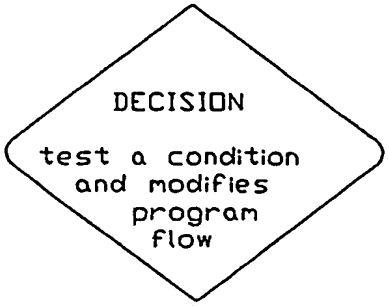
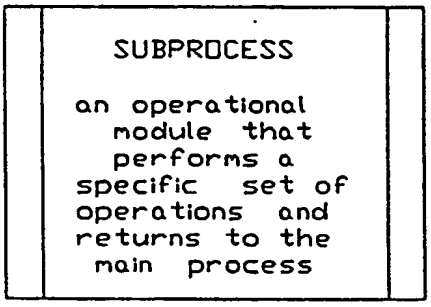
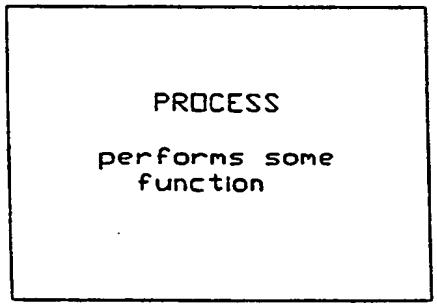


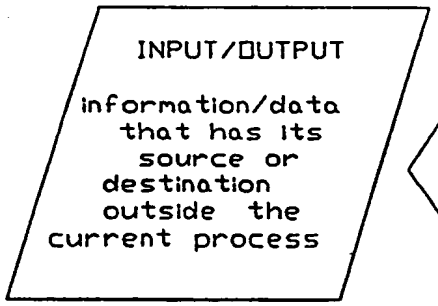
FIG. 2

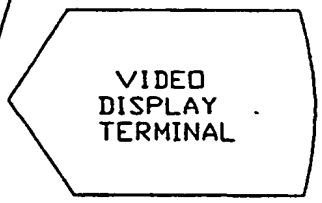


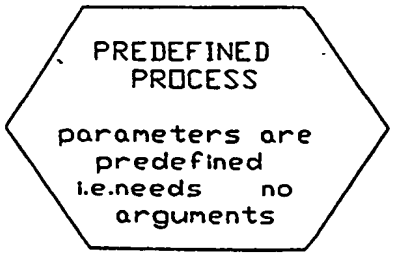
INDEX TO SYMBOLS USED IN FIGS. 3-21

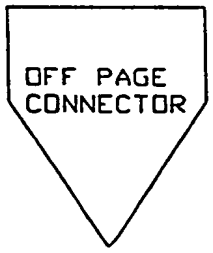


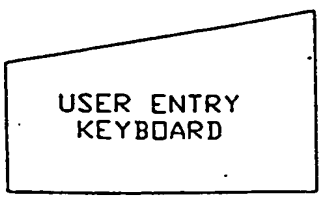












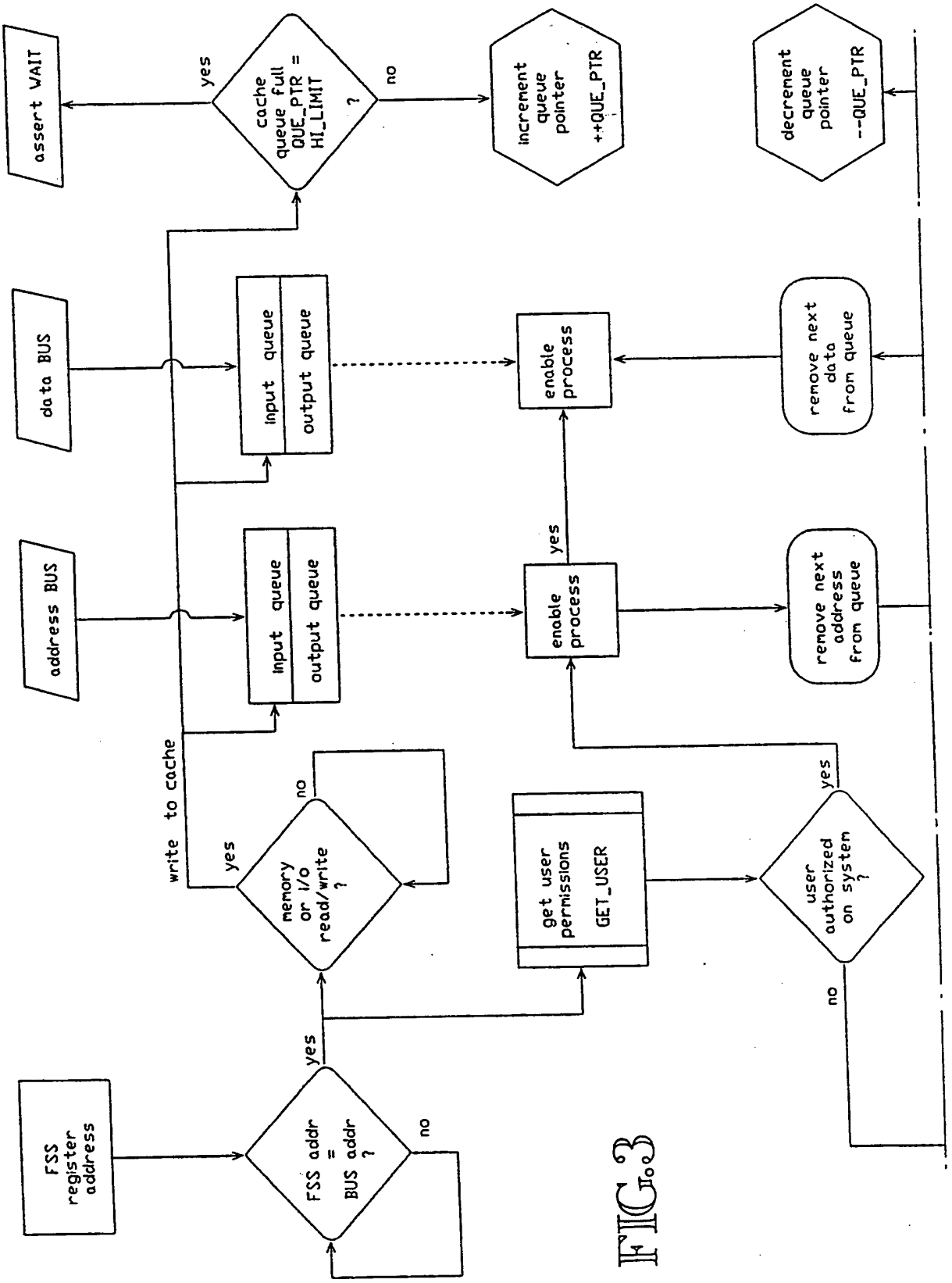
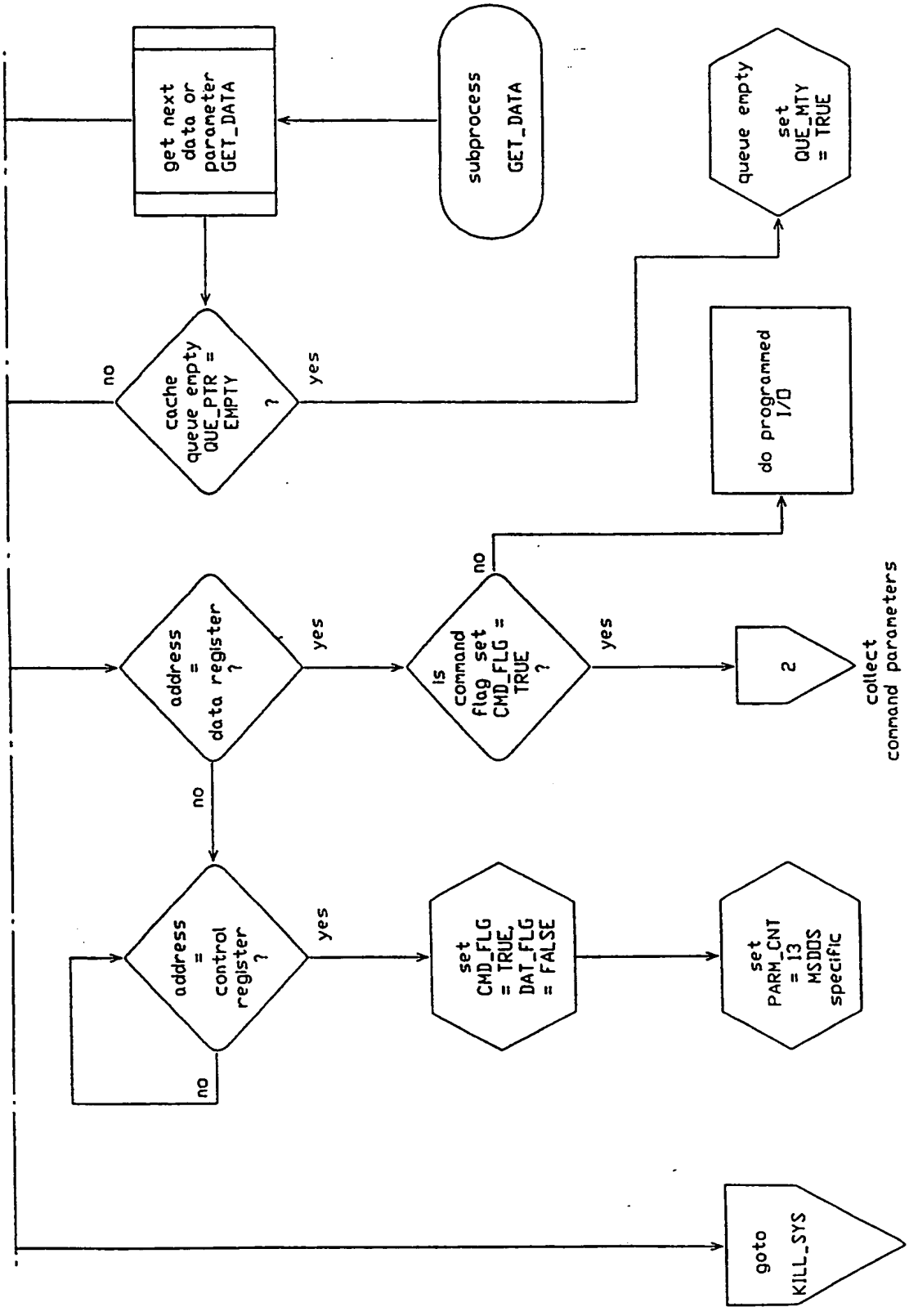

FSS register address
FSS addr = BUS addr ?
yes
no
memory or i/o read/write ?
yes
no
write to cache
get user permissions GET_USER
user authorized on system ?
yes
no
address BUS
input queue
output queue
yes
enable process
remove next address from queue
data BUS
input queue
output queue
enable process
remove next data from queue
cache queue full QUE_PTR = HI_LIMIT ?
yes
no
assert WAIT
increment queue pointer ++QUE_PTR
decrement queue pointer --QUE_PTR

FIG 3

FIG. 3 CONT.


get next data or parameter GET_DATA
subprocess GET_DATA
queue empty set QUE_MTY = TRUE
cache queue empty QUE_PTR = EMPTY ?
no
yes
do programmed I/O
address = data register ?
yes
no
is command flag set CMD_FLG = TRUE ?
yes
no
2
collect command parameters
address = control register ?
yes
no
set CMD_FLG = TRUE, DAT_FLG = FALSE
set PARM_CNT = 13 MSDOS specific
goto KILL_SYS

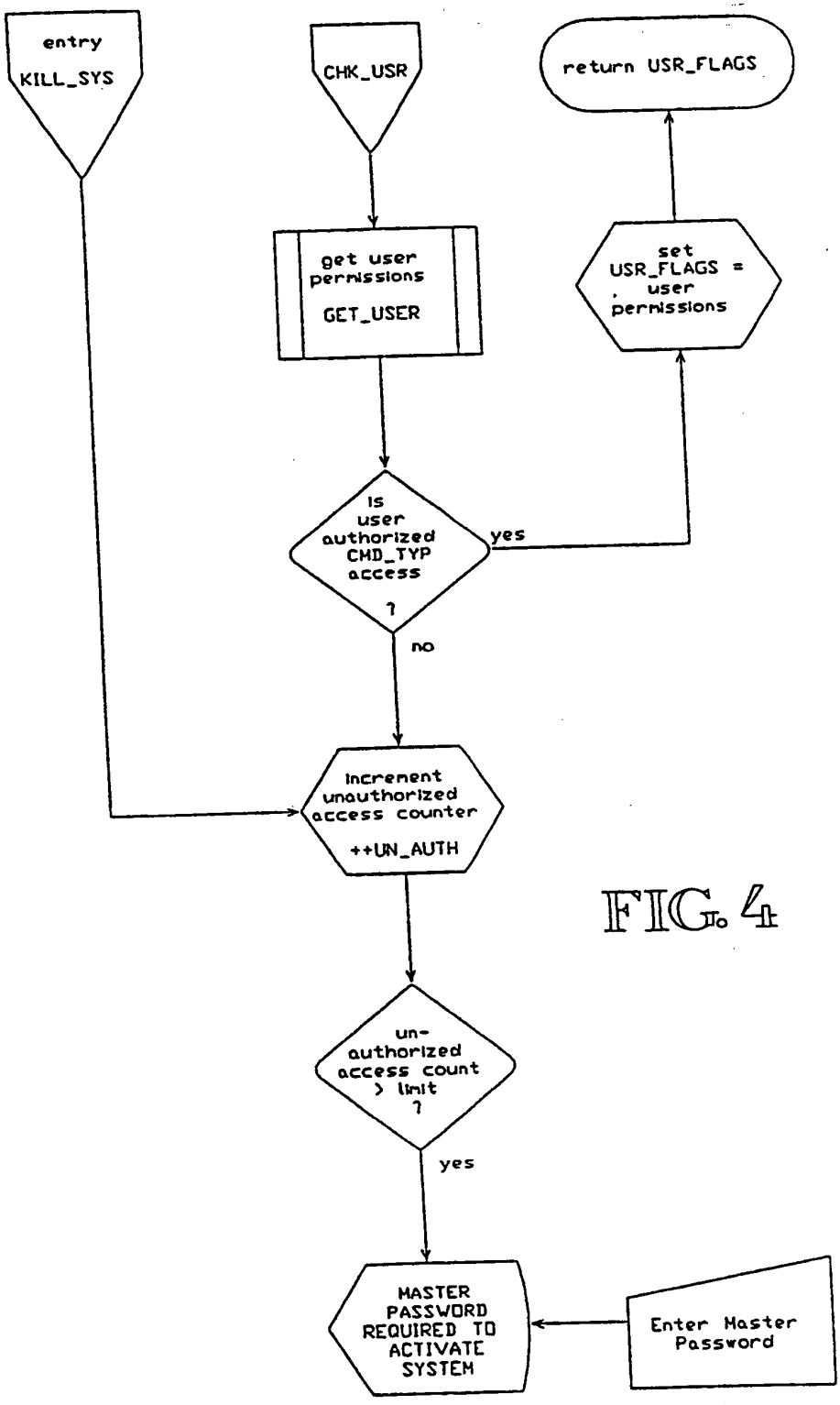

entry
KILL_SYS
CHK_USR
return USR_FLAGS
get user permissions
GET_USER
set USR_FLAGS = user permissions
Is user authorized CMD_TYP access ?
yes
no
Increment unauthorized access counter
++UN_AUTH
un-authorized access count > limit ?
yes
MASTER PASSWORD REQUIRED TO ACTIVATE SYSTEM
Enter Master Password

FIG. 4

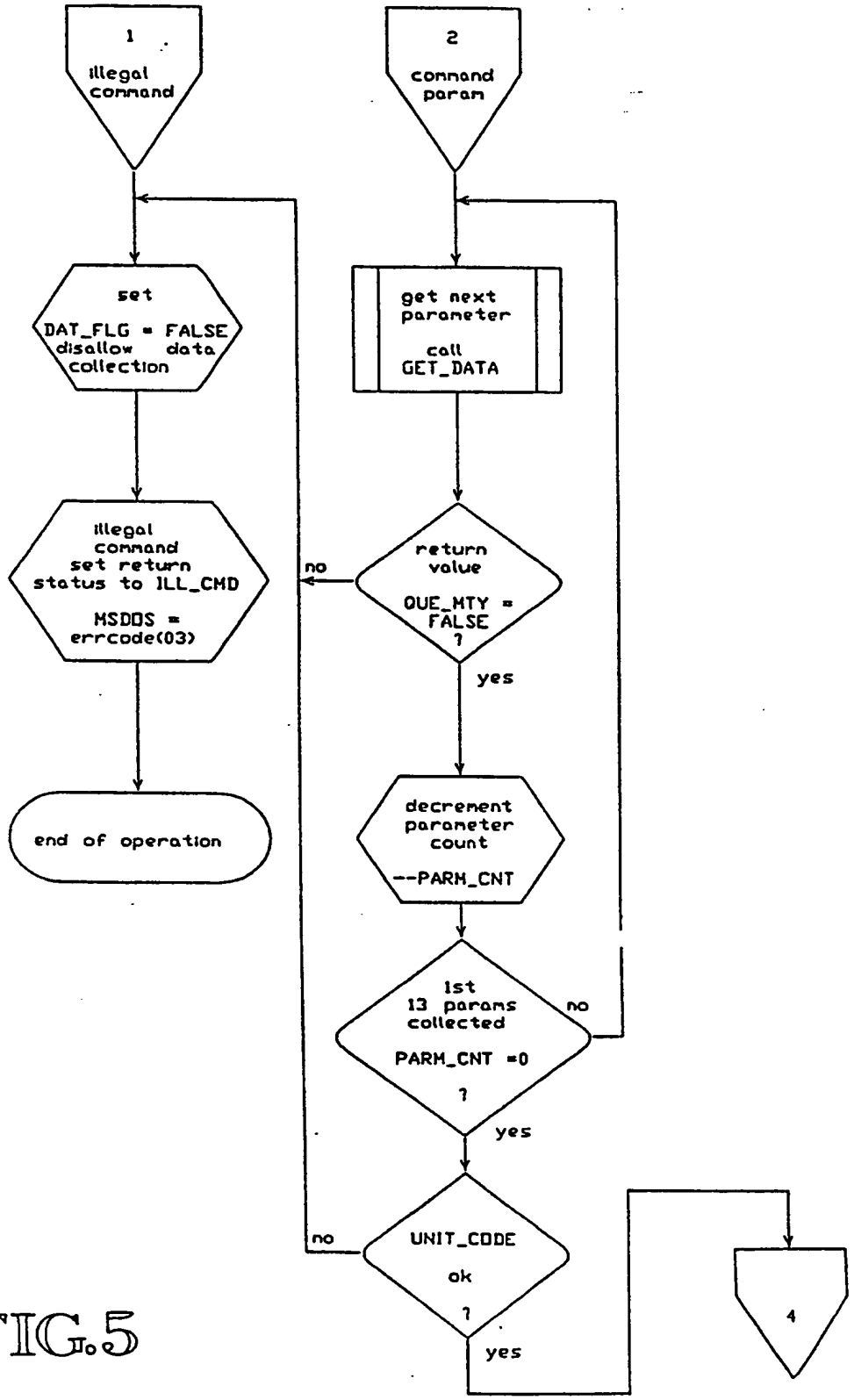
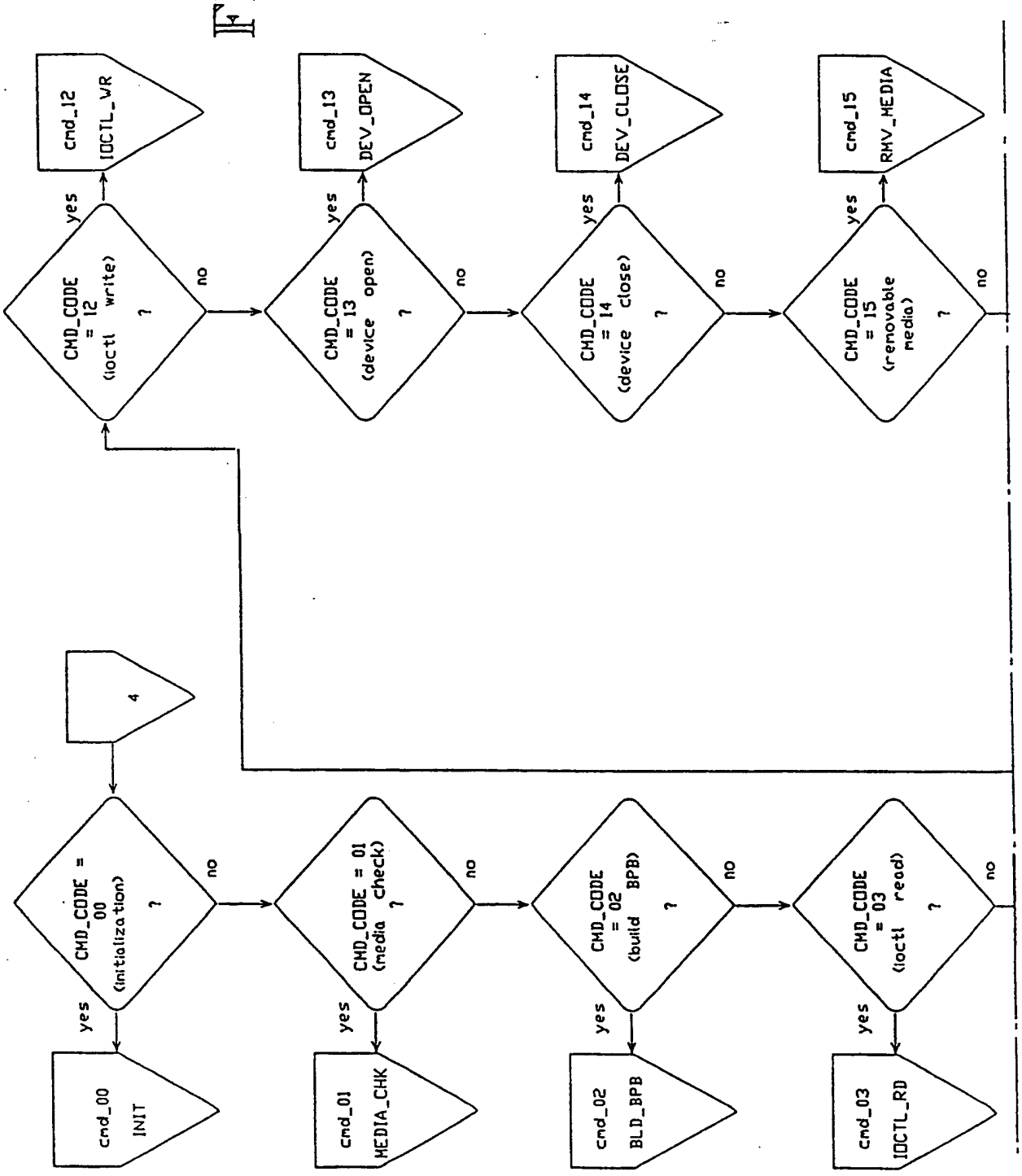

1
illegal command
2
command param
set DAT_FLG = FALSE disallow data collection
get next parameter call GET_DATA
illegal command set return status to ILL_CMD MSDOS = errcode(03)
return value QUE_MTY = FALSE ?
no
yes
end of operation
decrement parameter count --PARM_CNT
1st 13 params collected PARM_CNT =0 ?
no
yes
UNIT_CODE ok ?
no
yes
4

FIG.5

FIG.6


4
CMD_CODE = 00 (initialization) ?
yes
no
cmd_00 INIT
CMD_CODE = 01 (media check) ?
yes
no
cmd_01 MEDIA_CHK
CMD_CODE = 02 (build BPB) ?
yes
no
cmd_02 BLD_BPB
CMD_CODE = 03 (ioctl read) ?
yes
no
cmd_03 IOCTL_RD
CMD_CODE = 12 (ioctl write) ?
yes
no
cmd_12 IOCTL_WR
CMD_CODE = 13 (device open) ?
yes
no
cmd_13 DEV_OPEN
CMD_CODE = 14 (device close) ?
yes
no
cmd_14 DEV_CLOSE
CMD_CODE = 15 (removable media) ?
yes
no
cmd_15 RMV_MEDIA

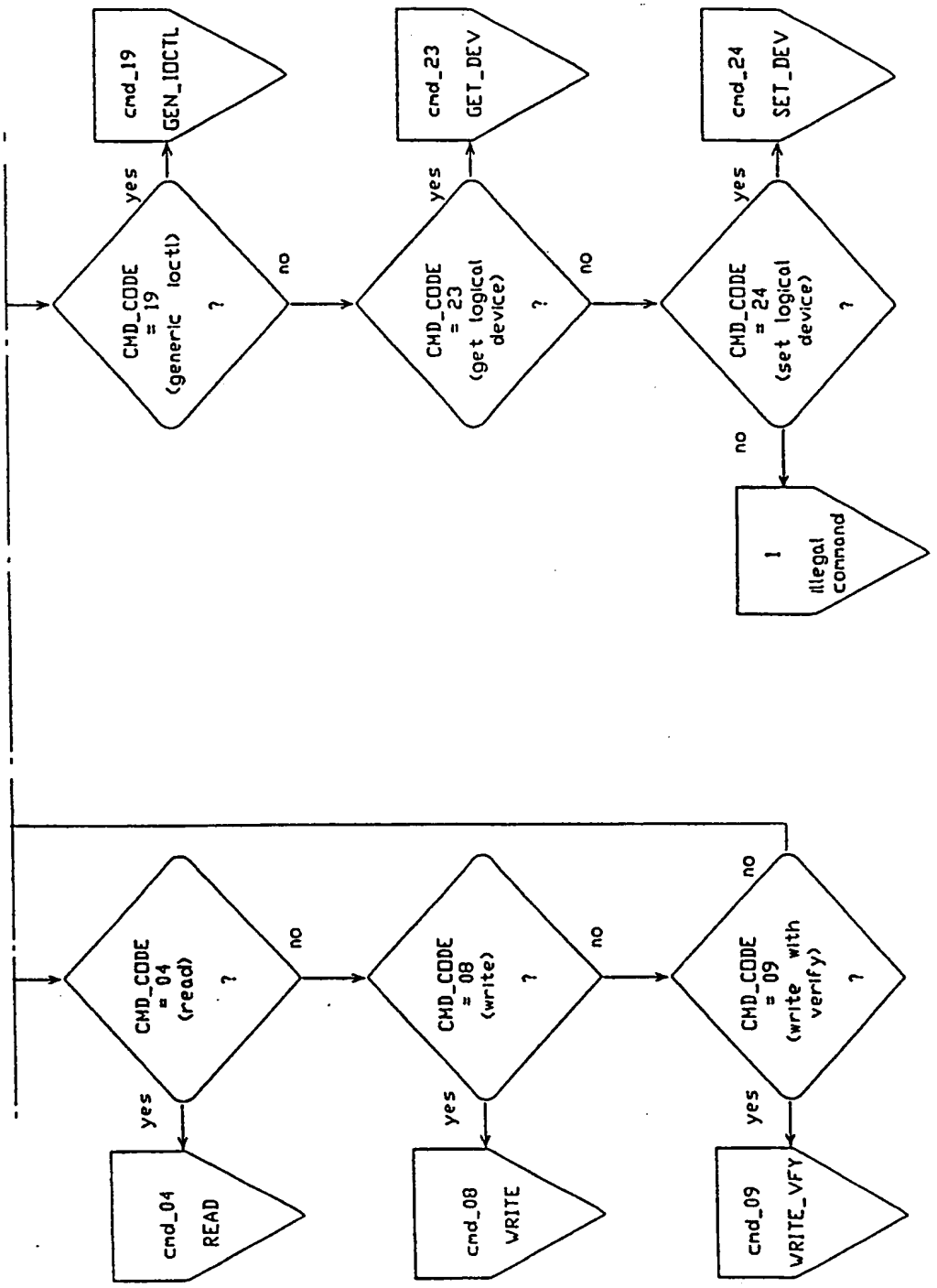

CMD_CODE = 19 (generic ioctl) ?
yes
cmd_19 GEN_IOCTL
no
CMD_CODE = 23 (get logical device) ?
yes
cmd_23 GET_DEV
no
CMD_CODE = 24 (set logical device) ?
yes
cmd_24 SET_DEV
no
1 illegal command
CMD_CODE = 04 (read) ?
yes
cmd_04 READ
no
CMD_CODE = 08 (write) ?
yes
cmd_08 WRITE
no
CMD_CODE = 09 (write with verify) ?
yes
cmd_09 WRITE_VFY
no

FIG. 6 CONT.

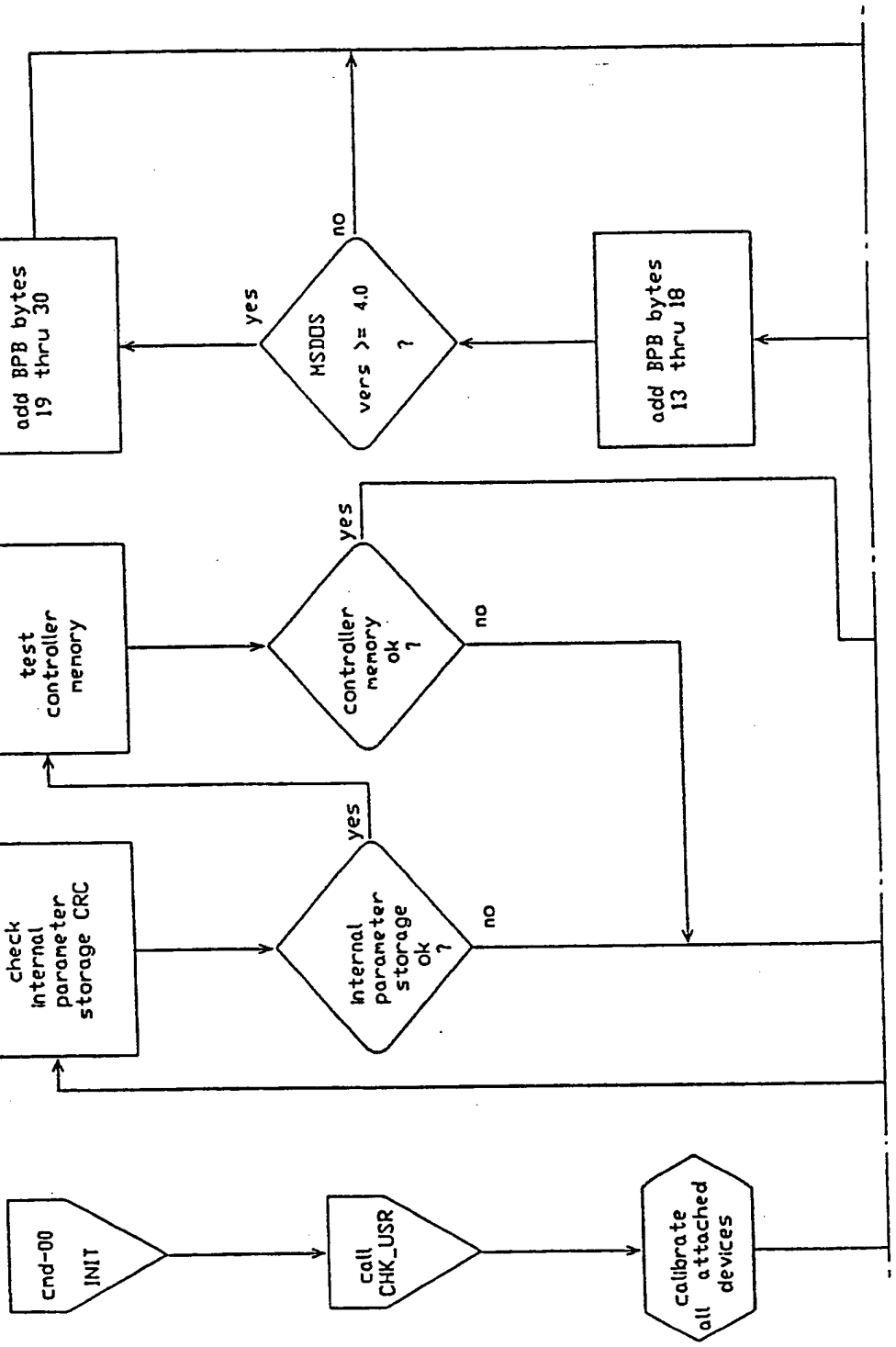

cnd-00
INIT
call
CHK_USR
calibrate
all attached
devices
check
internal
parameter
storage CRC
internal
parameter
storage
ok
?
yes
no
test
controller
memory
controller
memory
ok
?
yes
no
add BPB bytes
19 thru 30
MSDOS
vers >= 4.0
?
yes
no
add BPB bytes
13 thru 18

FIG. 7

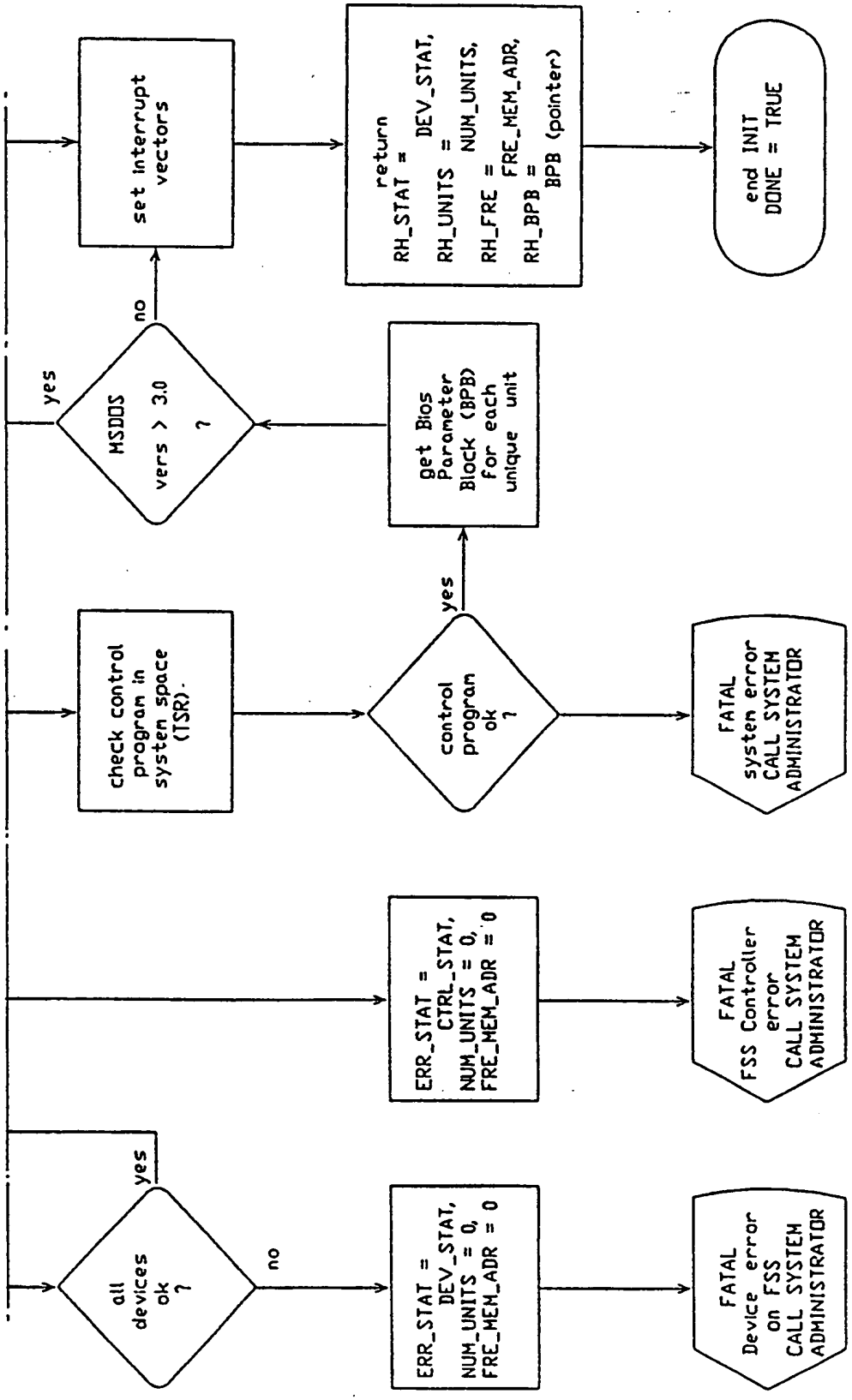


FIG. 7 CONT.

SUBSTITUTE SHEET

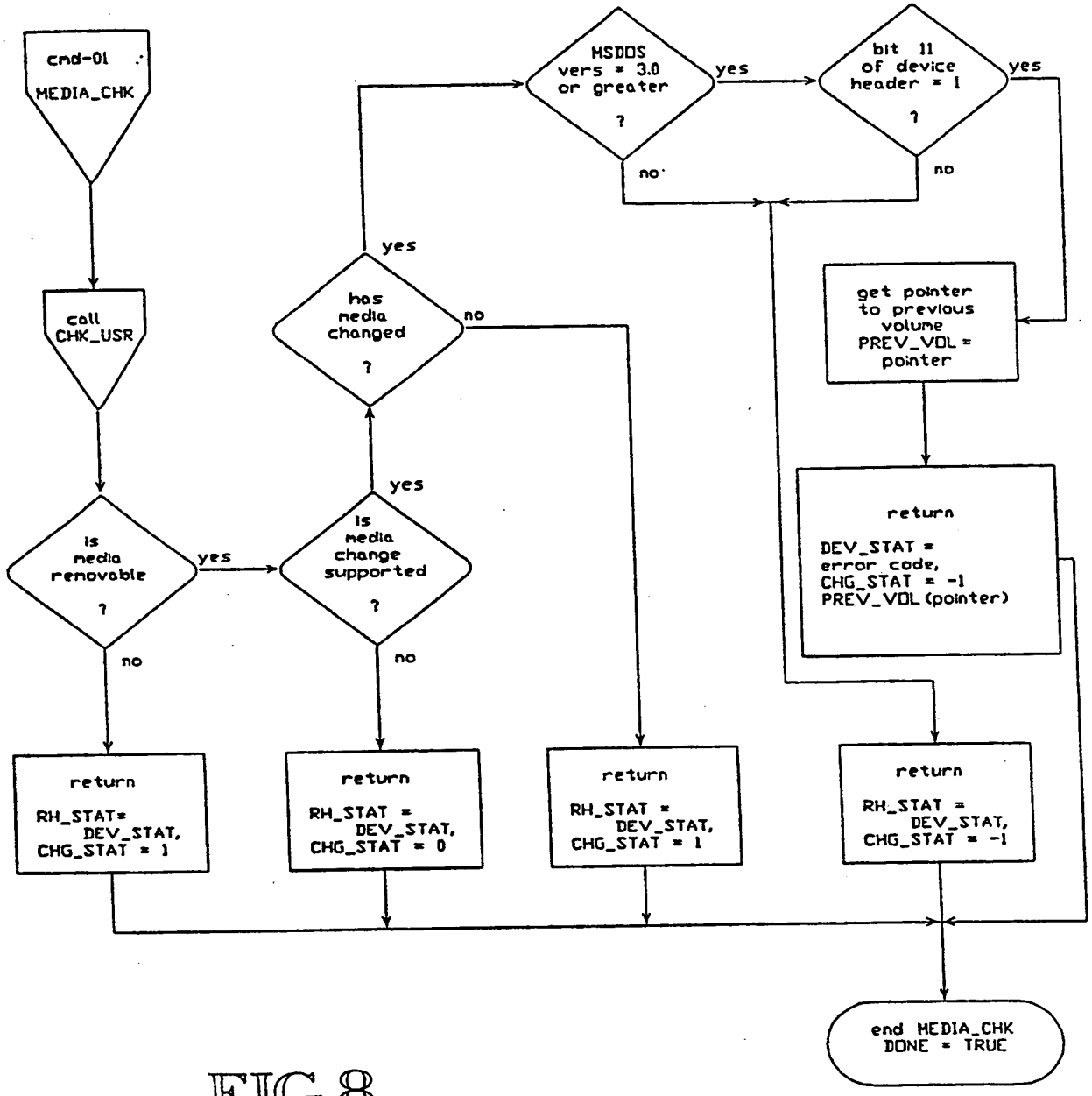
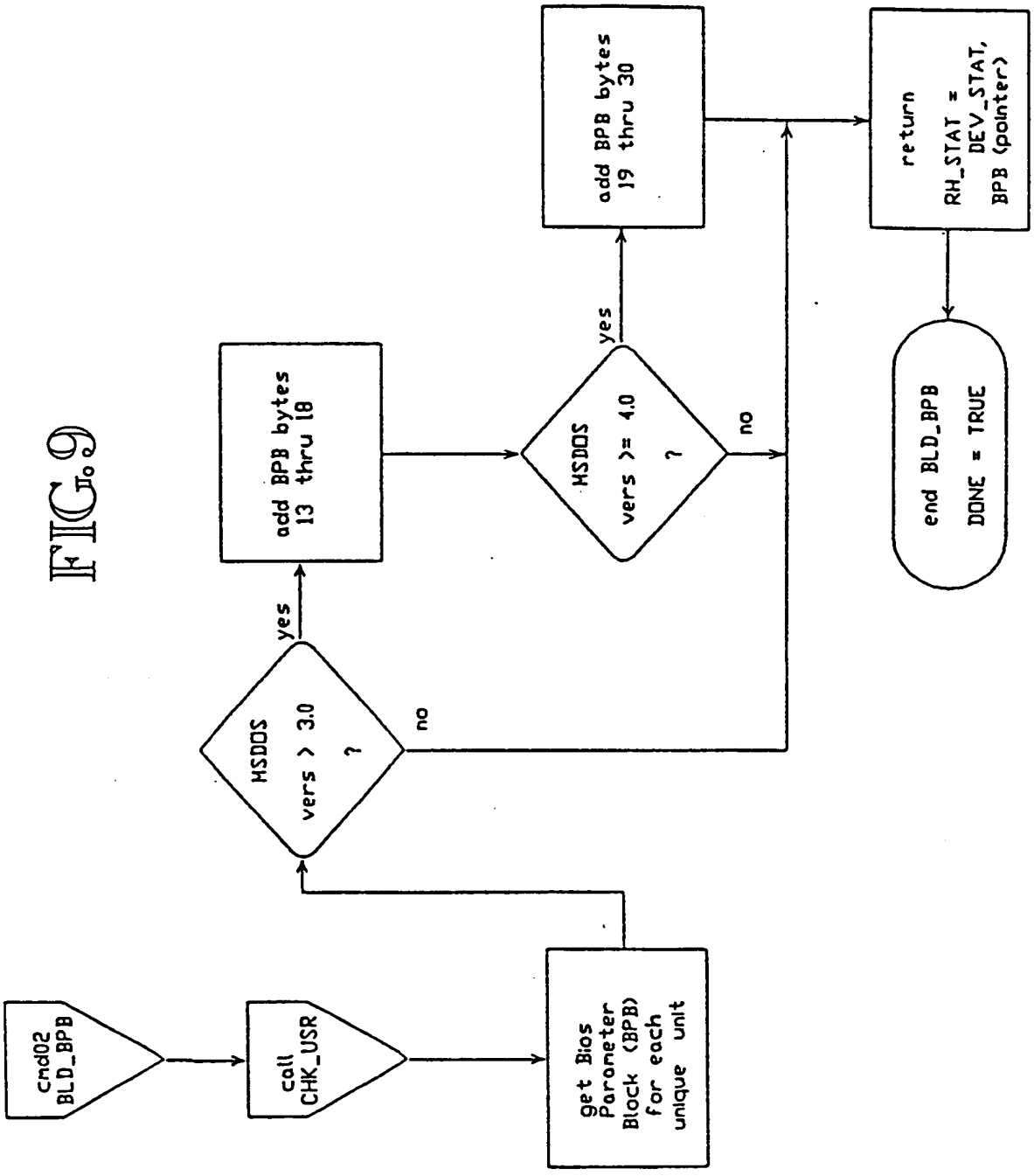

cmd-01
MEDIA_CHK
call
CHK_USR
is
media
removable
?
yes
no
return
RH_STAT=
DEV_STAT,
CHG_STAT = 1
is
media
change
supported
?
yes
no
has
media
changed
?
yes
no
return
RH_STAT =
DEV_STAT,
CHG_STAT = 0
MSDOS
vers = 3.0
or greater
?
yes
no
return
RH_STAT =
DEV_STAT,
CHG_STAT = 1
bit 11
of device
header = 1
?
yes
no
get pointer
to previous
volume
PREV_VOL =
pointer
return
DEV_STAT =
error code,
CHG_STAT = -1
PREV_VOL (pointer)
return
RH_STAT =
DEV_STAT,
CHG_STAT = -1
end MEDIA_CHK
DONE = TRUE

FIG. 8

FIG. 9


cmd02
BLD_BPB
call
CHK_USR
get Bios
Parameter
Block (BPB)
for each
unique unit
HSDOS
vers > 3.0
?
yes
no
add BPB bytes
13 thru 18
HSDOS
vers >= 4.0
?
yes
no
add BPB bytes
19 thru 30
return
RH_STAT =
DEV_STAT,
BPB (pointer)
end BLD_BPB
DONE = TRUE

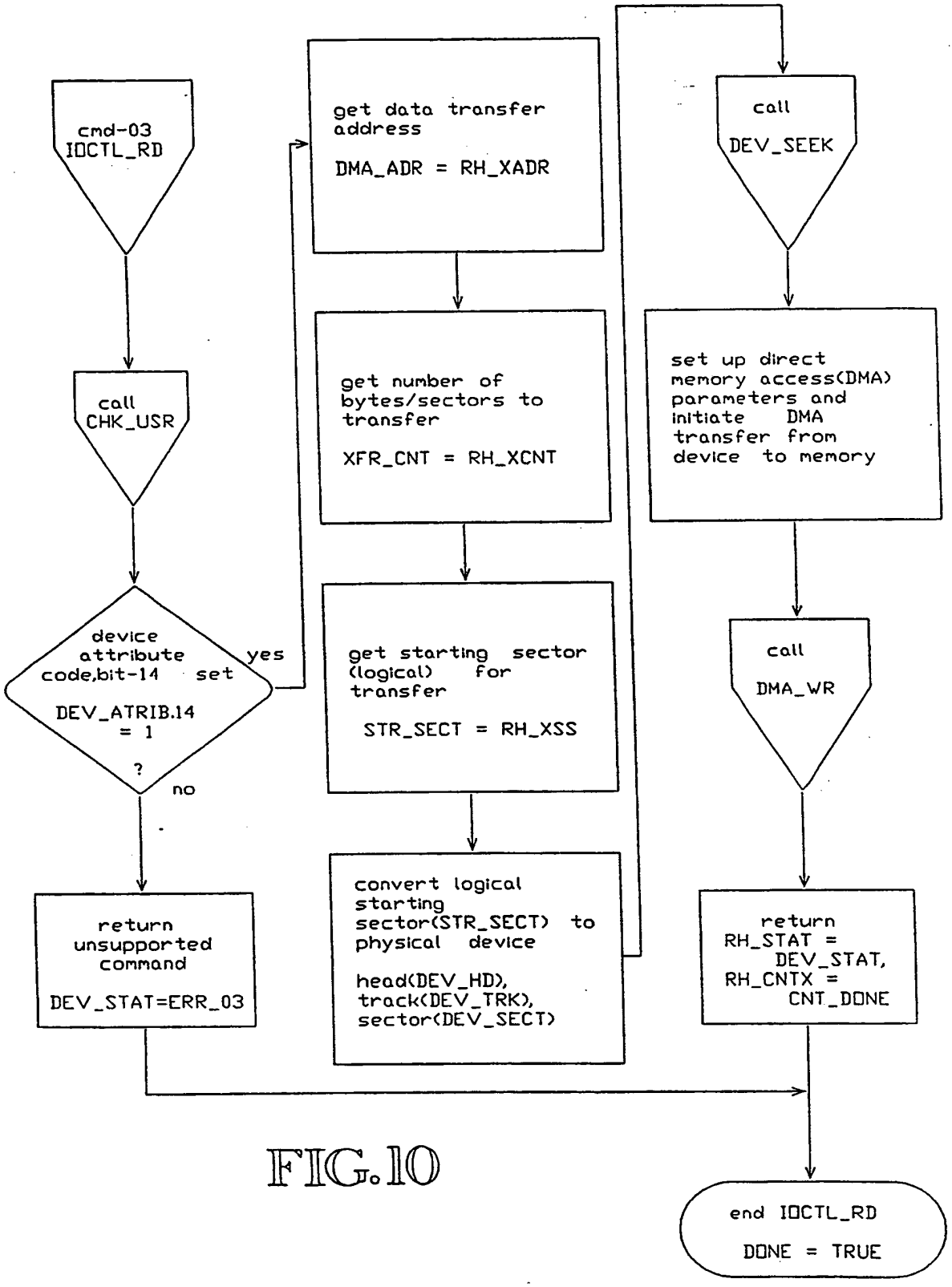

cmd-03
IOCTL_RD
call
CHK_USR
device
attribute
code,bit-14 set
DEV_ATRIB.14
= 1
?
yes
no
return
unsupported
command
DEV_STAT=ERR_03
get data transfer
address
DMA_ADR = RH_XADR
get number of
bytes/sectors to
transfer
XFR_CNT = RH_XCNT
get starting sector
(logical) for
transfer
STR_SECT = RH_XSS
convert logical
starting
sector(STR_SECT) to
physical device
head(DEV_HD),
track(DEV_TRK),
sector(DEV_SECT)
call
DEV_SEEK
set up direct
memory access(DMA)
parameters and
initiate DMA
transfer from
device to memory
call
DMA_WR
return
RH_STAT =
DEV_STAT,
RH_CNTX =
CNT_DONE
end IOCTL_RD
DONE = TRUE

FIG. 10

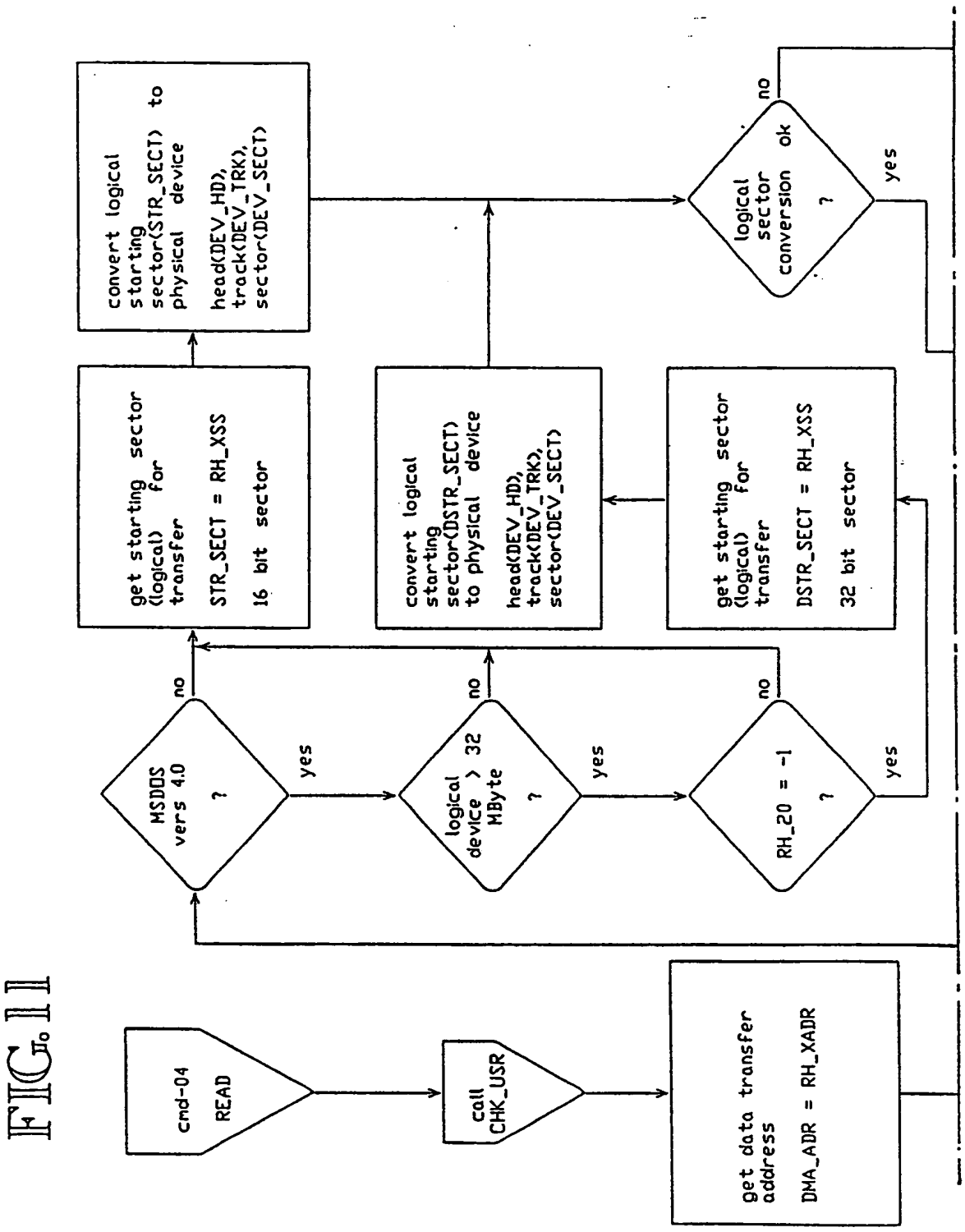

cmd-04
READ
call
CHK_USR
get data transfer
address
DMA_ADR = RH_XADR
MSDOS
vers 4.0
?
no
yes
logical
device > 32
MByte
?
no
yes
RH_20 = -1
?
no
yes
get starting sector
(logical) for
transfer
STR_SECT = RH_XSS
16 bit sector
convert logical
starting
sector(STR_SECT) to
physical device
head(DEV_HD),
track(DEV_TRK),
sector(DEV_SECT)
convert logical
starting
sector(DSTR_SECT)
to physical device
head(DEV_HD),
track(DEV_TRK),
sector(DEV_SECT)
get starting sector
(logical) for
transfer
DSTR_SECT = RH_XSS
32 bit sector
logical
sector
conversion ok
?
no
yes

FIG. 11

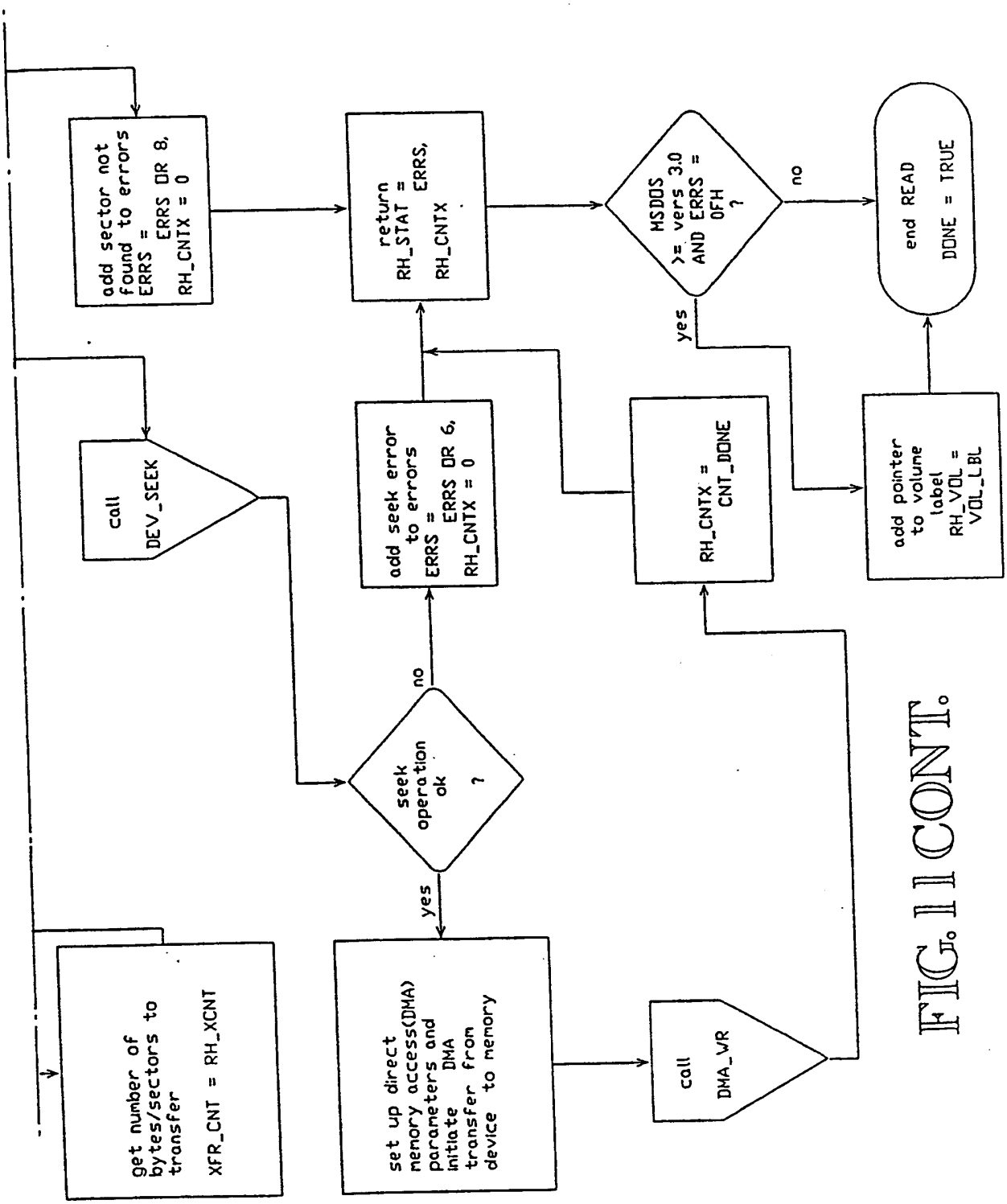


FIG. 11 CONT.

SUBSTITUTE SHEET

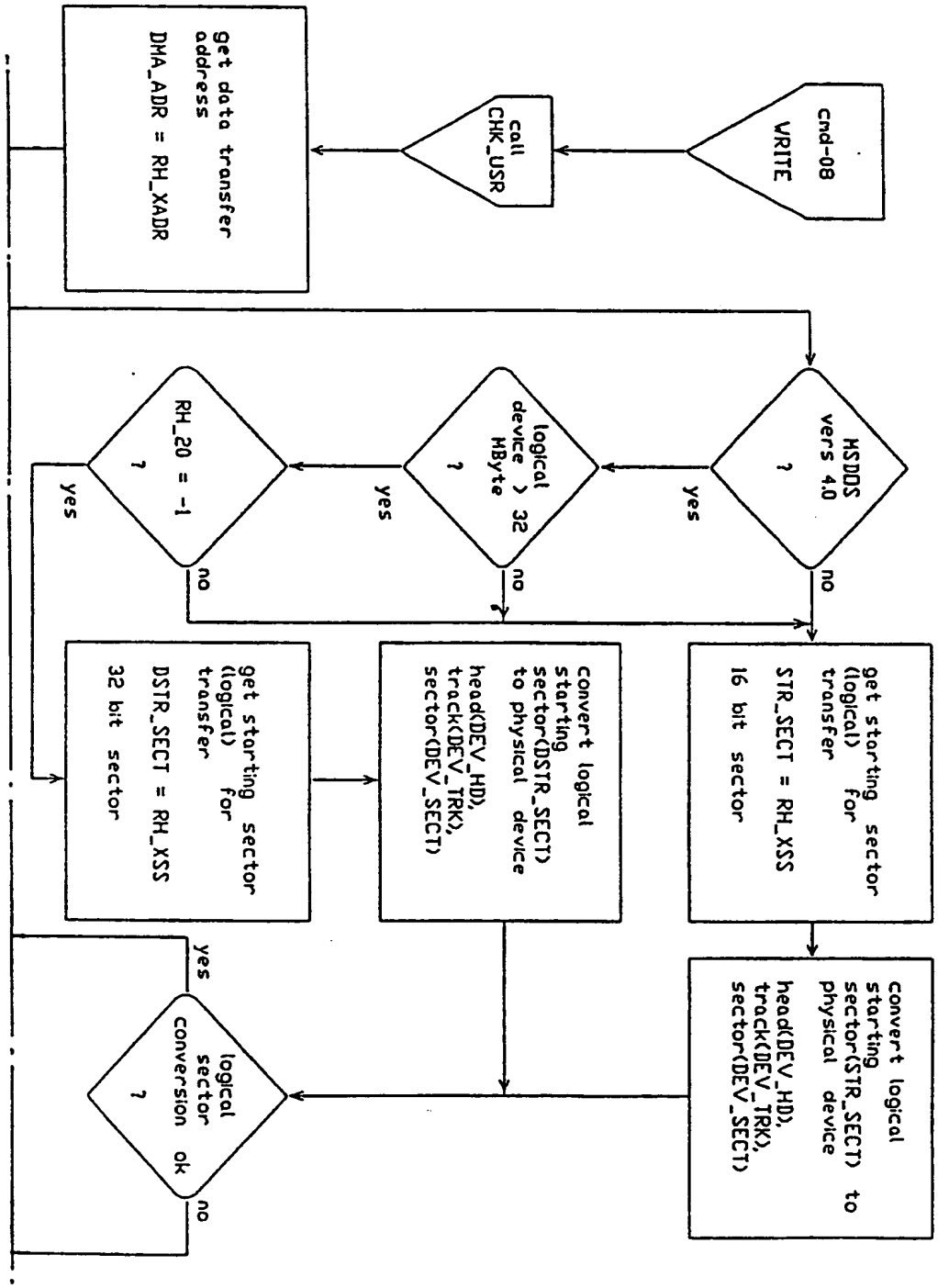

cmd-08 WRITE
call CHK_USR
get data transfer address DMA_ADR = RH_XADR
MSDOS vers 4.0 ?
yes
no
logical device > 32 MByte ?
yes
no
RH_20 = -1 ?
yes
no
get starting sector (logical) for transfer STR_SECT = RH_XSS 16 bit sector
convert logical starting sector(STR_SECT) to physical device head(DEV_HD), track(DEV_TRK), sector(DEV_SECT)
get starting sector (logical) for transfer DSTR_SECT = RH_XSS 32 bit sector
convert logical starting sector(DSTR_SECT) to physical device head(DEV_HD), track(DEV_TRK), sector(DEV_SECT)
logical sector conversion ok ?
yes
no

FIG. 12

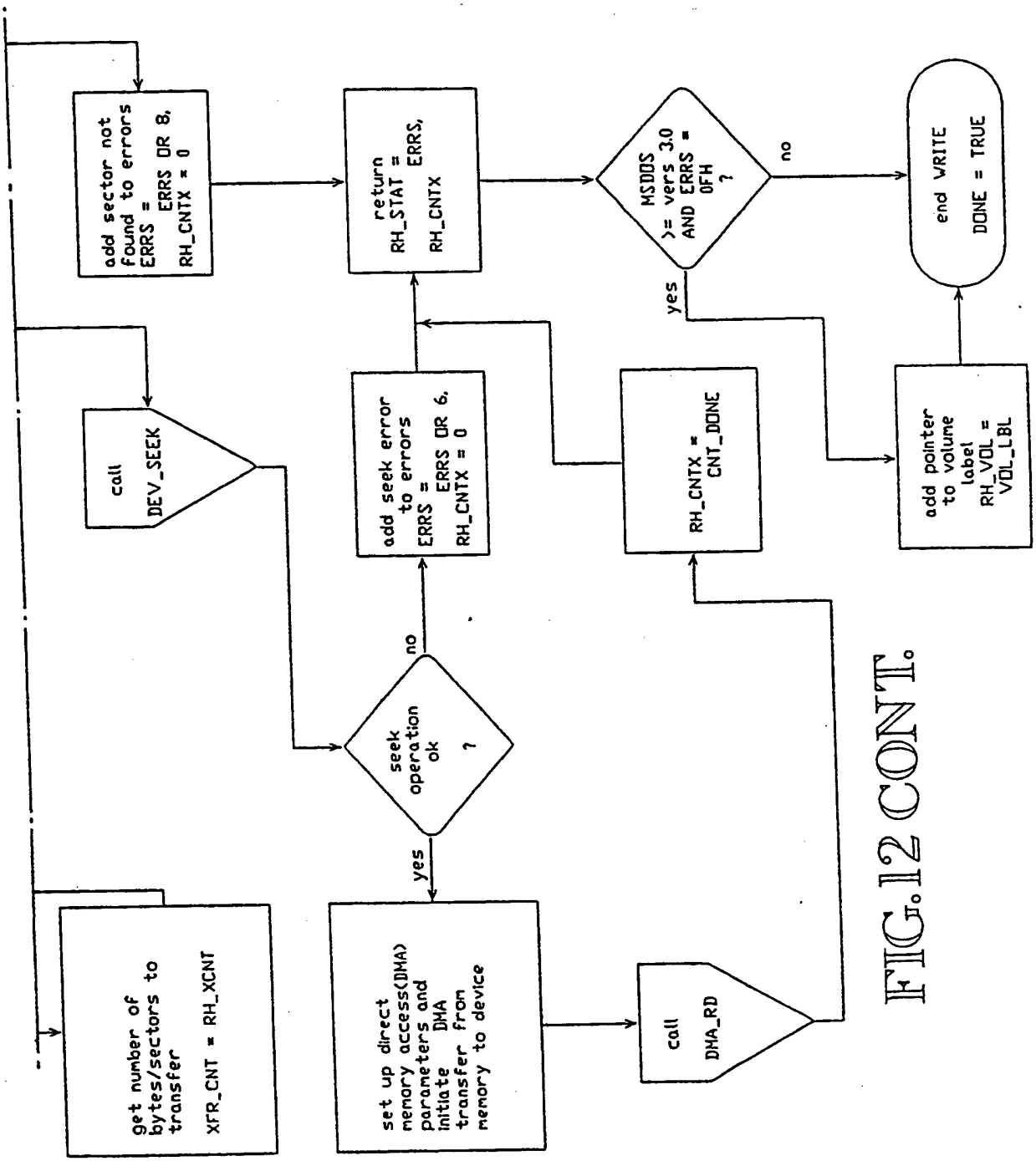

add sector not found to errors
ERRS = ERRS OR 8,
RH_CNTX = 0
return
RH_STAT = ERRS,
RH_CNTX
MSDOS >= vers 3.0 AND ERRS = 0FH ?
no
end WRITE
DONE = TRUE
yes
call
DEV_SEEK
add seek error to errors
ERRS = ERRS OR 6,
RH_CNTX = 0
RH_CNTX = CNT_DONE
add pointer to volume label
RH_VOL = VOL_LBL
no
seek operation ok ?
yes
get number of bytes/sectors to transfer
XFR_CNT = RH_XCNT
set up direct memory access(DMA) parameters and initiate DMA transfer from memory to device
call
DMA_RD

FIG. 12 CONT.

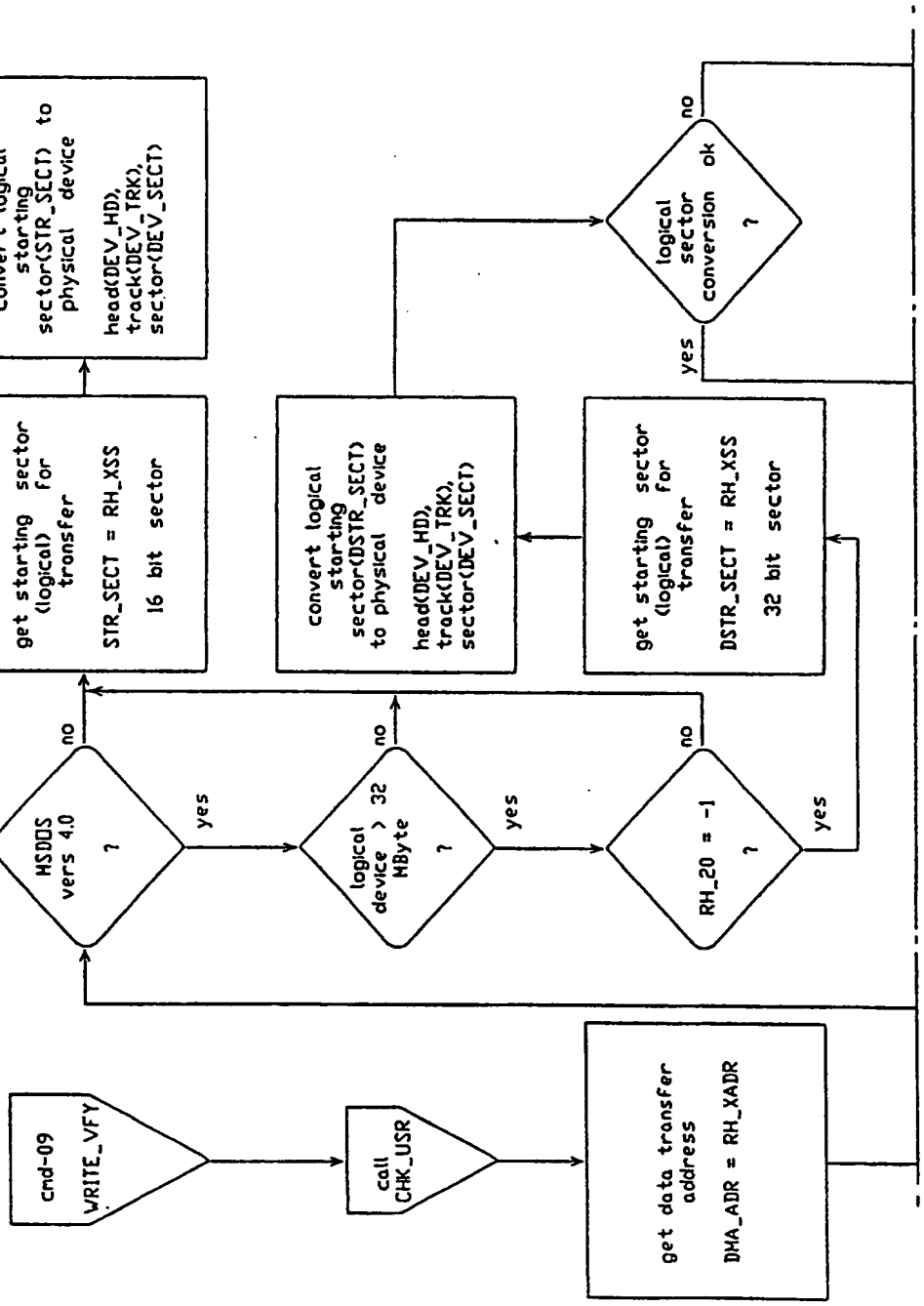
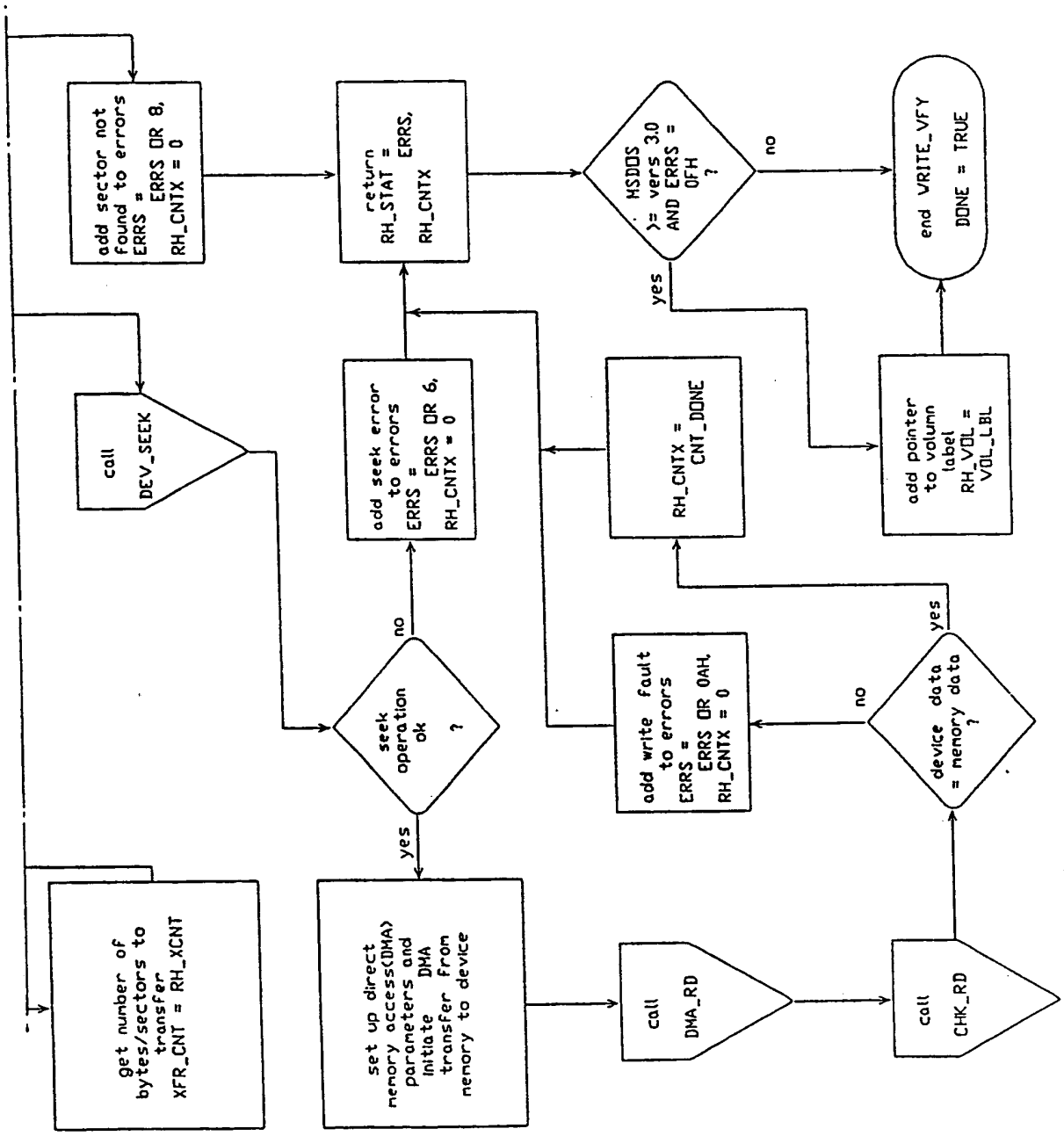

cmd-09
WRITE_VFY
call
CHK_USR
get data transfer
address
DMA_ADR = RH_XADR
MSDOS
vers 4.0
?
yes
no
logical
device > 32
MByte
?
yes
no
RH_20 = -1
?
yes
no
get starting sector
(logical)
transfer
DSTR_SECT = RH_XSS
32 bit sector
convert logical
starting
sector(DSTR_SECT)
to physical device
head(DEV_HD),
track(DEV_TRK),
sector(DEV_SECT)
logical
sector
conversion ok
?
yes
no
get starting sector
(logical)
transfer
STR_SECT = RH_XSS
16 bit sector
convert logical
starting
sector(STR_SECT) to
physical device
head(DEV_HD),
track(DEV_TRK),
sector(DEV_SECT)

FIG 13

FIG. 13 CONT.


add sector not found to errors
ERRS = ERRS OR 8,
RH_CNTX = 0
return
RH_STAT = ERRS,
RH_CNTX
MSDOS >= vers 3.0 AND ERRS = 0FH ?
no
end VRITE_VFY
DONE = TRUE
yes
call
DEV_SEEK
add seek error to errors
ERRS = ERRS OR 6,
RH_CNTX = 0
RH_CNTX = CNT_DONE
add pointer to volumn label
RH_VOL = VOL_LBL
no
seek operation ok ?
yes
add write fault to errors
ERRS = ERRS OR 0AH,
RH_CNTX = 0
no
device data = memory data ?
yes
get number of bytes/sectors to transfer
XFR_CNT = RH_XCNT
set up direct memory access(DMA) paraneters and initiate DMA transfer from memory to device
call
DMA_RD
call
CHK_RD

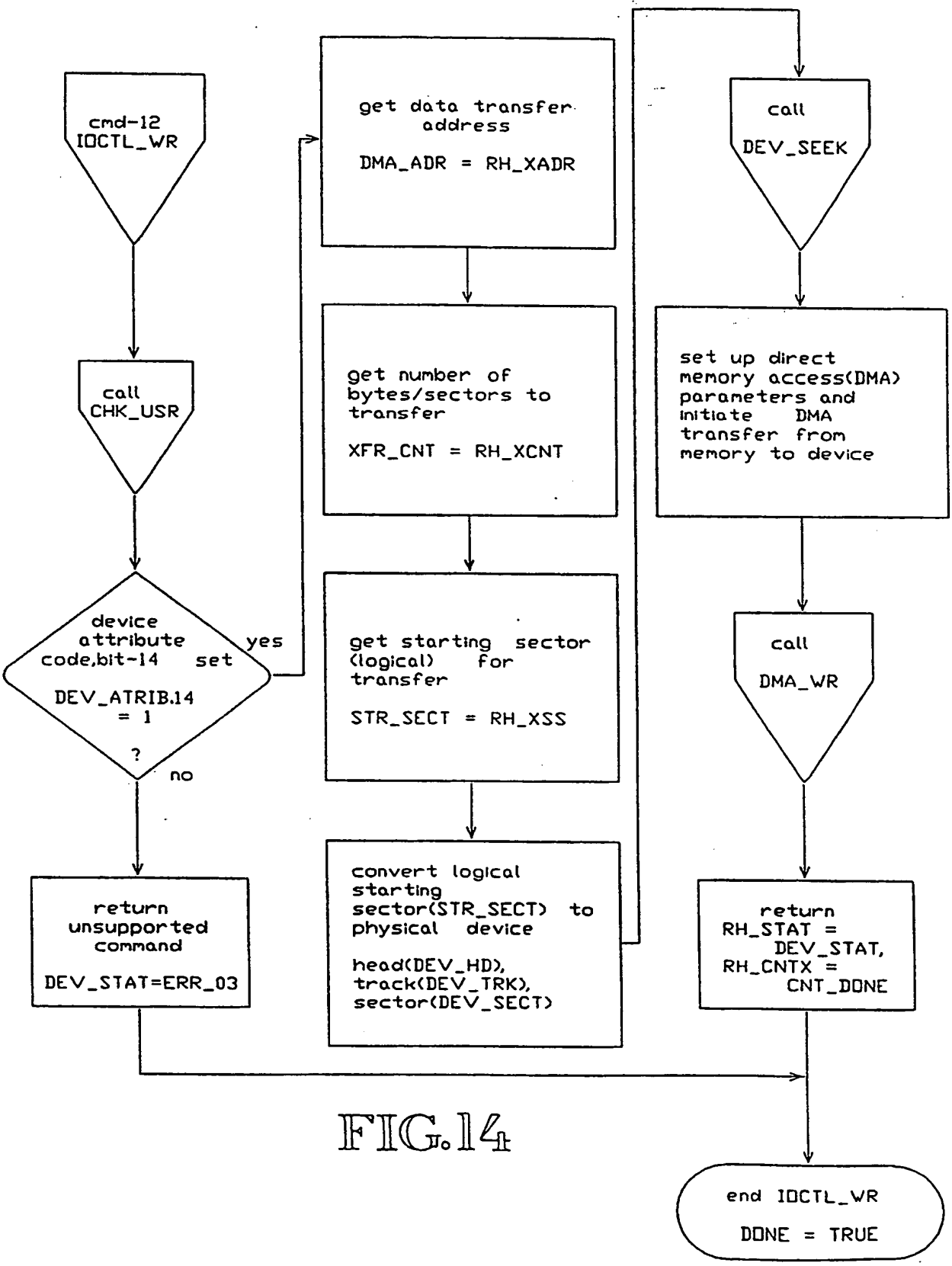
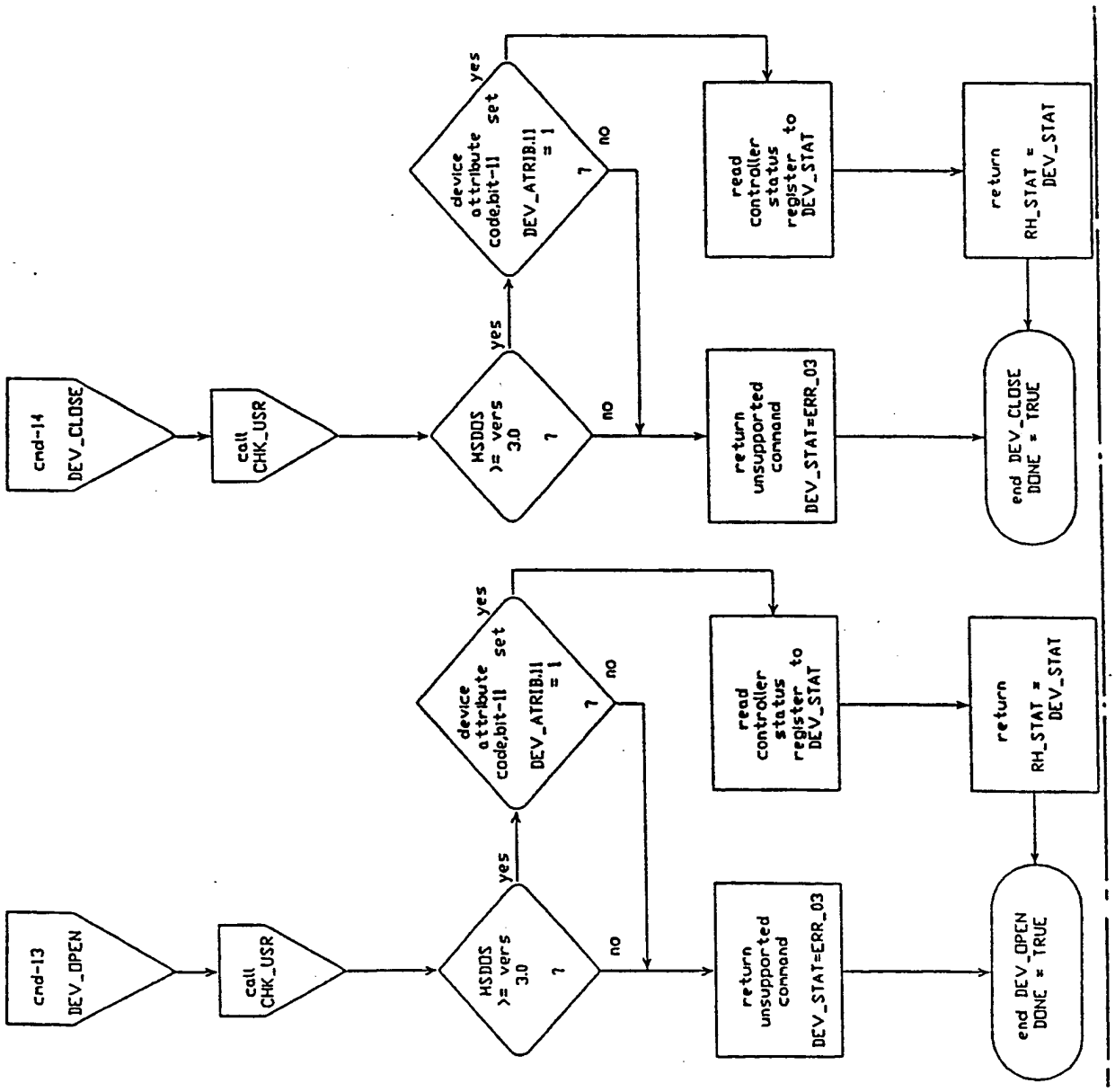

cmd-12
IOCTL_WR
call
CHK_USR
device
attribute
code,bit-14 set
DEV_ATRIB.14
= 1
?
yes
no
return
unsupported
command
DEV_STAT=ERR_03
get data transfer
address
DMA_ADR = RH_XADR
get number of
bytes/sectors to
transfer
XFR_CNT = RH_XCNT
get starting sector
(logical) for
transfer
STR_SECT = RH_XSS
convert logical
starting
sector(STR_SECT) to
physical device
head(DEV_HD),
track(DEV_TRK),
sector(DEV_SECT)
call
DEV_SEEK
set up direct
memory access(DMA)
parameters and
initiate DMA
transfer from
memory to device
call
DMA_WR
return
RH_STAT =
DEV_STAT,
RH_CNTX =
CNT_DONE
end IOCTL_WR
DONE = TRUE

FIG. 14

FIG. 15


cmd-13
DEV_OPEN
call
CHK_USR
HSDOS
>= vers
3.0
?
yes
no
device
attribute
code,bit-11
set
DEV_ATRIB.11
= 1
?
yes
no
return
unsupported
command
DEV_STAT=ERR_03
read
controller
status
register to
DEV_STAT
return
RH_STAT =
DEV_STAT
end DEV_OPEN
DONE = TRUE
cmd-14
DEV_CLOSE
call
CHK_USR
HSDOS
>= vers
3.0
?
yes
no
device
attribute
code,bit-11
set
DEV_ATRIB.11
= 1
?
yes
no
return
unsupported
command
DEV_STAT=ERR_03
read
controller
status
register to
DEV_STAT
return
RH_STAT =
DEV_STAT
end DEV_CLOSE
DONE = TRUE

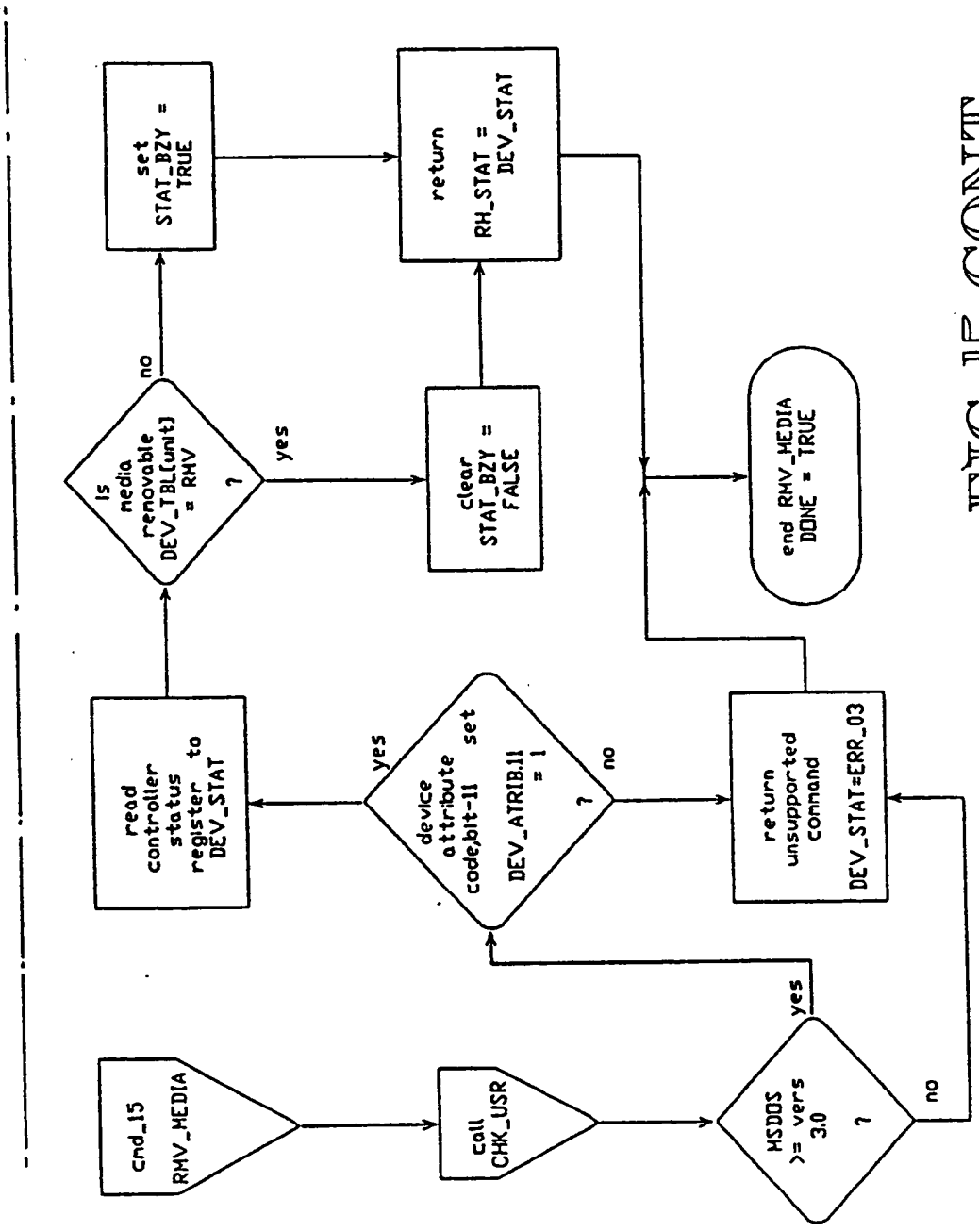

cmd_15
RMV_MEDIA
call
CHK_USR
MSDOS
>= vers
3.0
?
yes
no
device
attribute
code,bit-11
DEV_ATRIB.11
= 1
?
yes
no
read
controller
status
register to
DEV_STAT
return
unsupported
command
DEV_STAT=ERR_03
Is
media
removable
DEV_TBL[unit]
= RMV
?
no
yes
set
STAT_BZY =
TRUE
clear
STAT_BZY =
FALSE
return
RH_STAT =
DEV_STAT
end RMV_MEDIA
DONE = TRUE

FIG. 15 CONT.

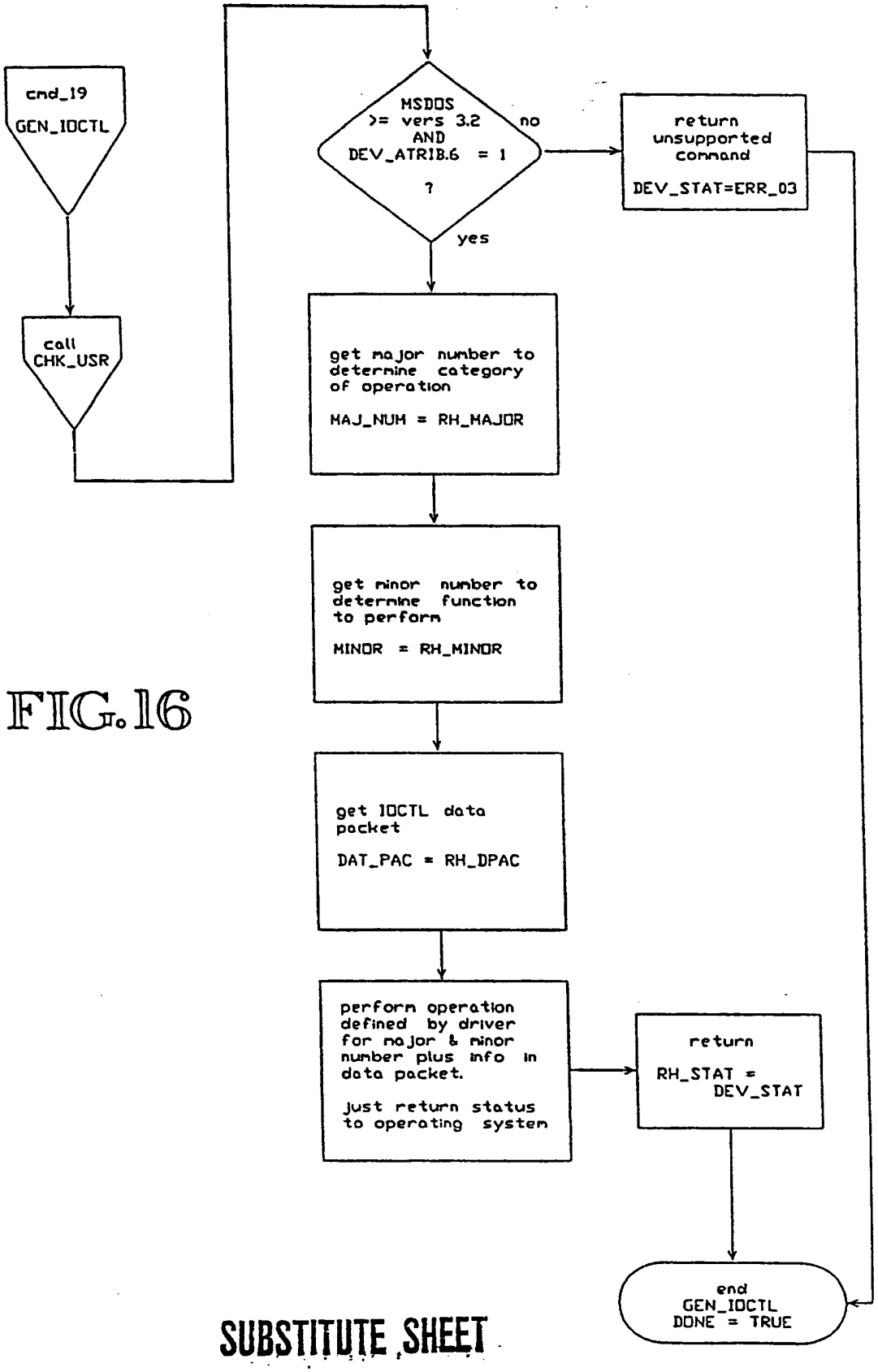

cmd_19
GEN_IOCTL
call
CHK_USR
MSDOS
>= vers 3.2
AND
DEV_ATRIB.6 = 1
?
no
yes
return
unsupported
command
DEV_STAT=ERR_03
get major number to
determine category
of operation
MAJ_NUM = RH_MAJOR
get minor number to
determine function
to perform
MINOR = RH_MINOR
get IOCTL data
packet
DAT_PAC = RH_DPAC
perform operation
defined by driver
for major & minor
number plus info in
data packet.
just return status
to operating system
return
RH_STAT =
DEV_STAT
end
GEN_IOCTL
DONE = TRUE

FIG. 16

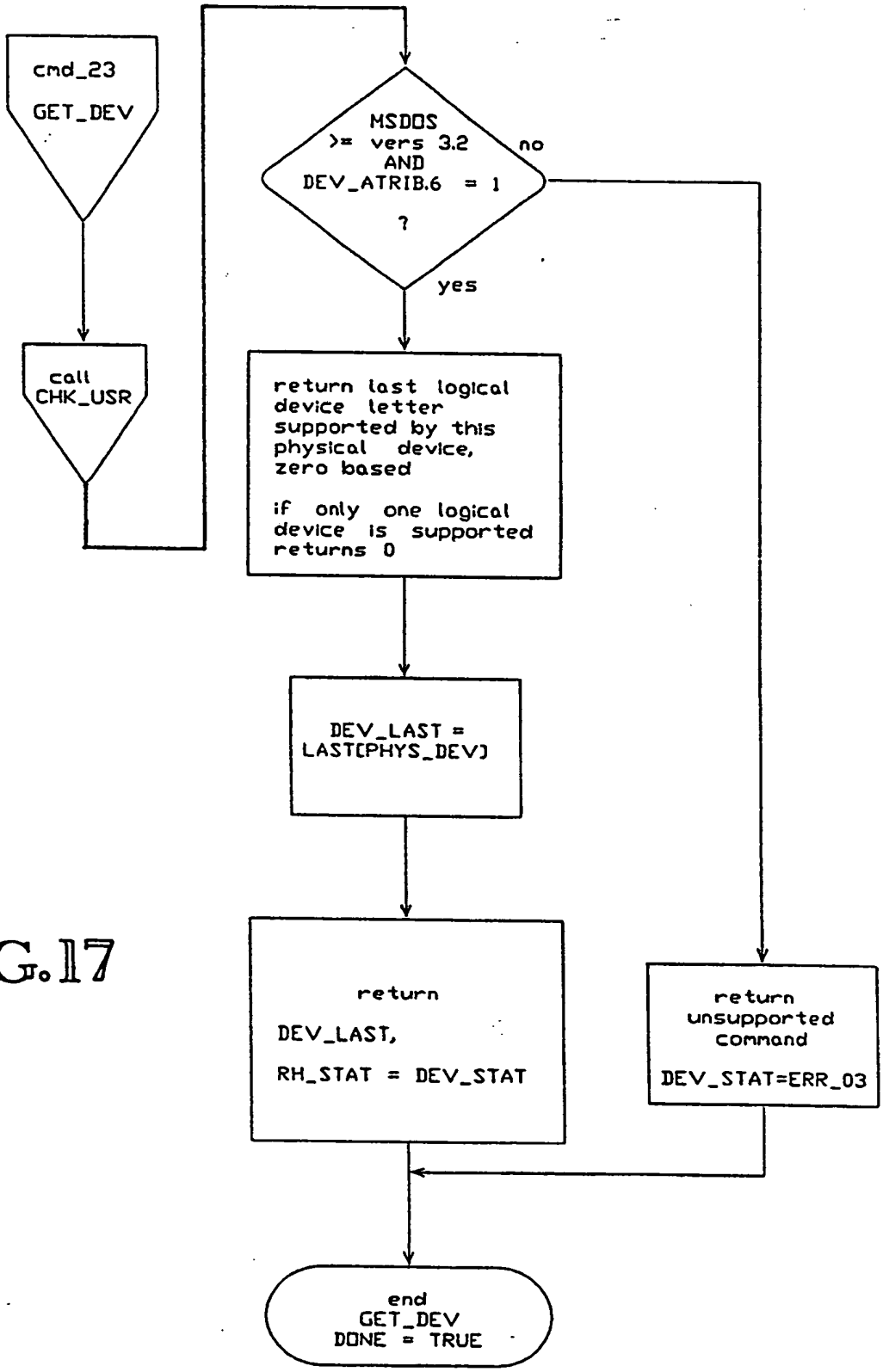

cmd_23
GET_DEV
call
CHK_USR
MSDOS
>= vers 3.2
AND
DEV_ATRIB.6 = 1
?
no
yes
return last logical device letter supported by this physical device, zero based
if only one logical device is supported returns 0
DEV_LAST = LAST[PHYS_DEV]
return
DEV_LAST,
RH_STAT = DEV_STAT
return unsupported command
DEV_STAT=ERR_03
end
GET_DEV
DONE = TRUE

FIG. 17

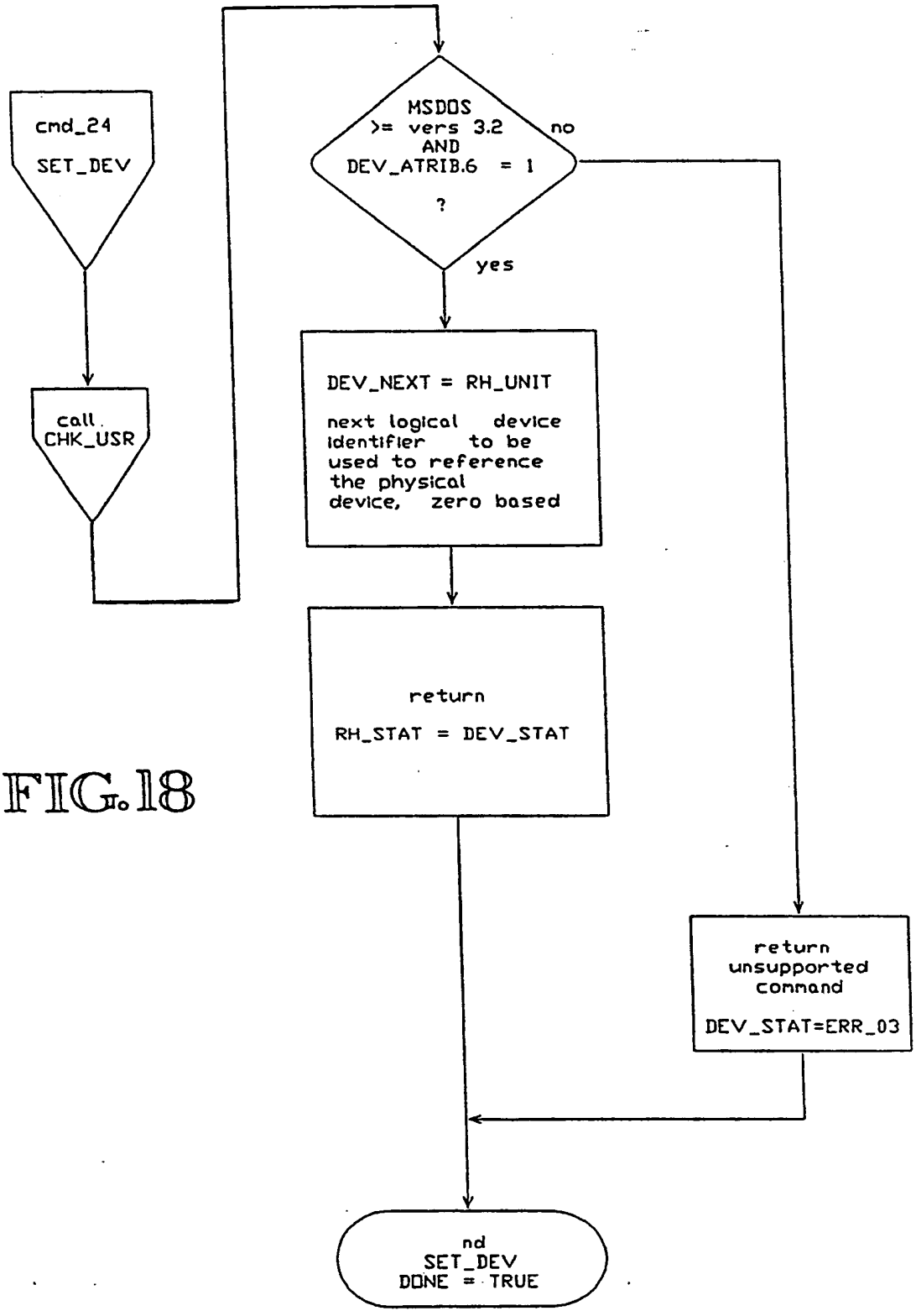

cmd_24
SET_DEV
call
CHK_USR
MSDOS
>= vers 3.2
AND
DEV_ATRIB.6 = 1
?
no
yes
DEV_NEXT = RH_UNIT
next logical device identifier to be used to reference the physical device, zero based
return
RH_STAT = DEV_STAT
return
unsupported
command
DEV_STAT=ERR_03
nd
SET_DEV
DONE = TRUE

FIG. 18

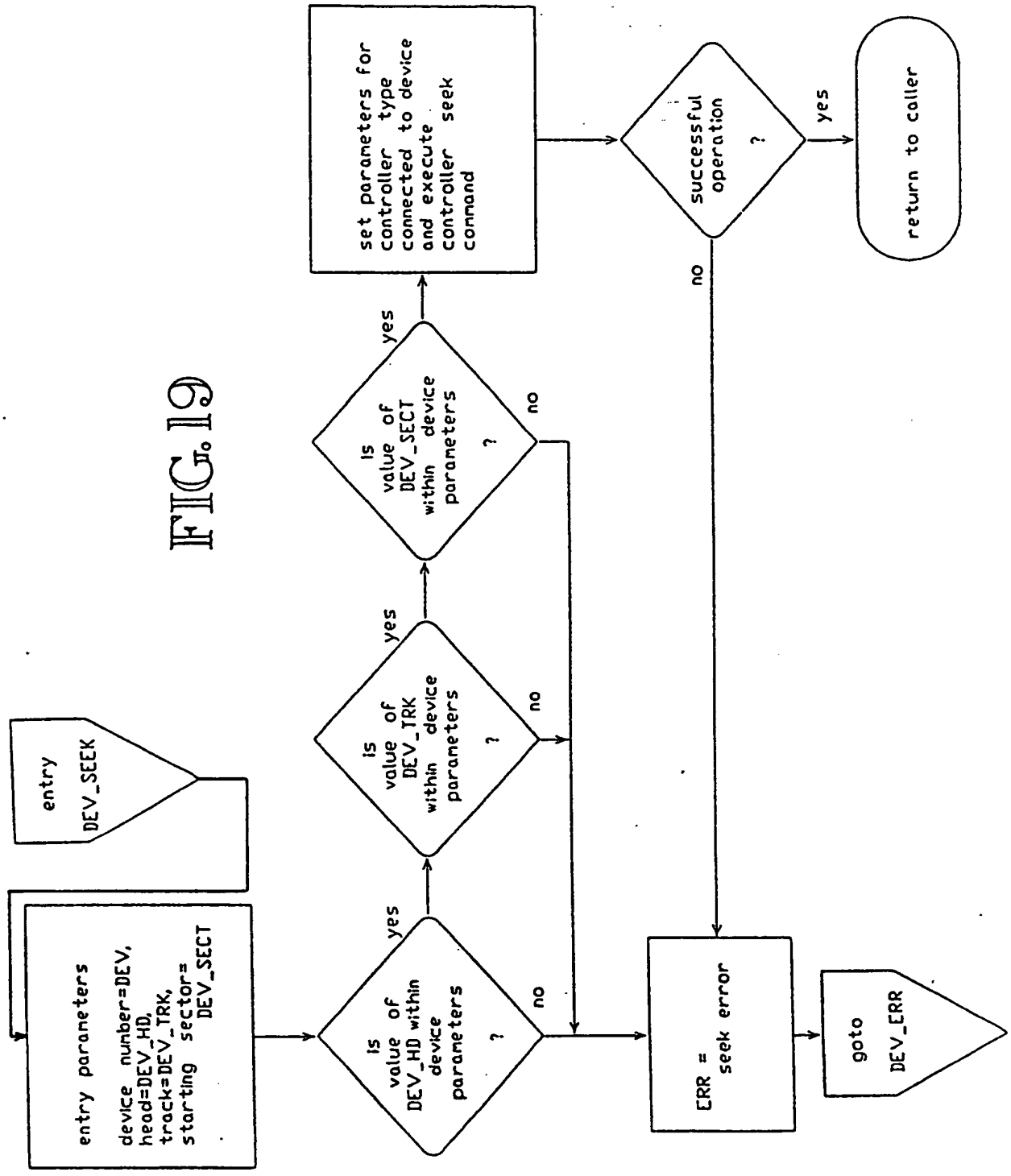

entry
DEV_SEEK
entry paraneters
device number=DEV,
head=DEV_HD,
track=DEV_TRK,
starting sector=
DEV_SECT
is value of DEV_HD within device parameters ?
yes
no
is value of DEV_TRK within device parameters ?
yes
no
is value of DEV_SECT within device parameters ?
yes
no
set parameters for controller type connected to device and execute controller seek command
successful operation ?
yes
no
return to caller
ERR = seek error
goto DEV_ERR

FIG. 19

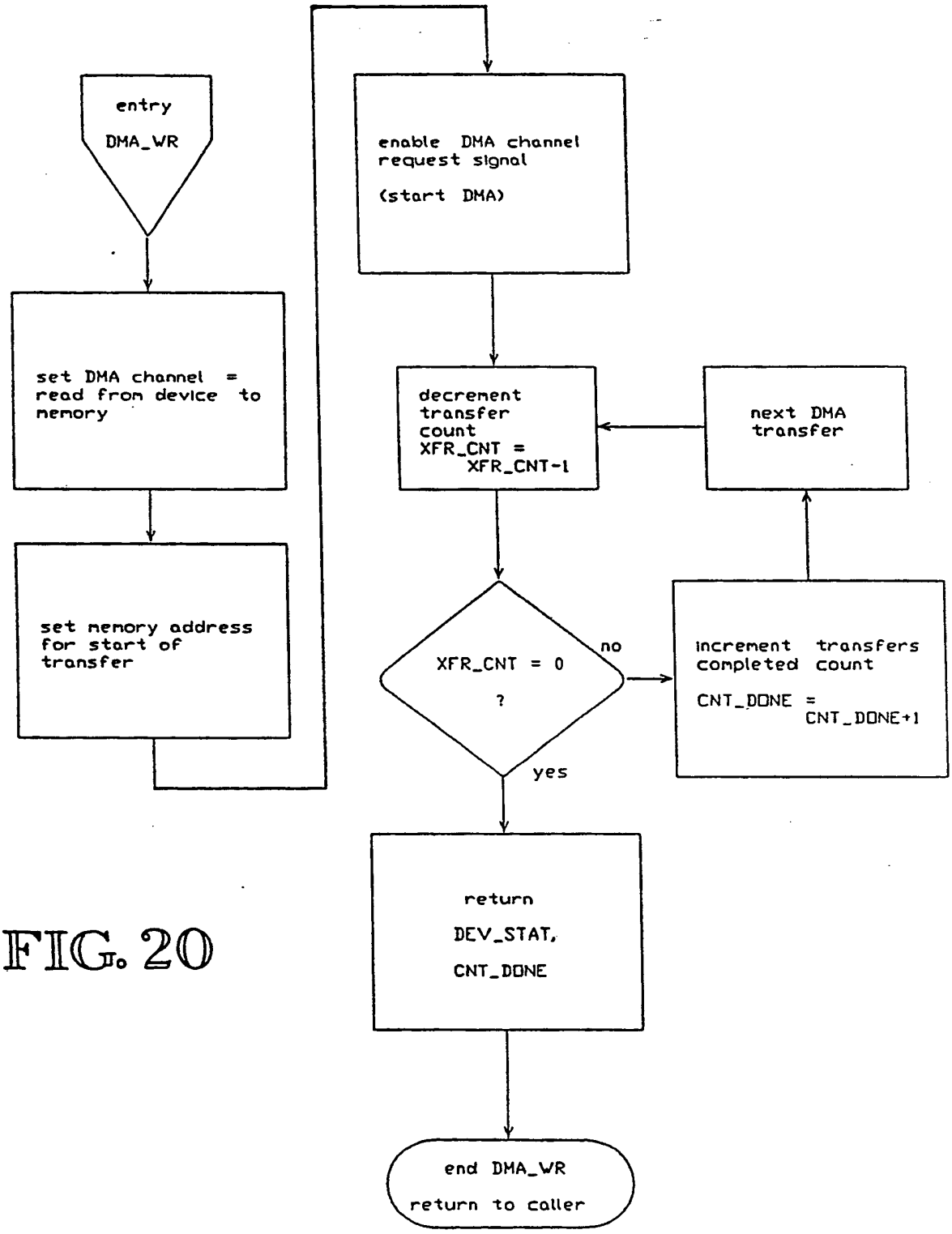

entry
DMA_WR
set DMA channel = read from device to memory
set memory address for start of transfer
enable DMA channel request signal
(start DMA)
decrement transfer count
XFR_CNT = XFR_CNT-1
next DMA transfer
XFR_CNT = 0 ?
no
yes
increment transfers completed count
CNT_DONE = CNT_DONE+1
return
DEV_STAT,
CNT_DONE
end DMA_WR
return to caller

FIG. 20

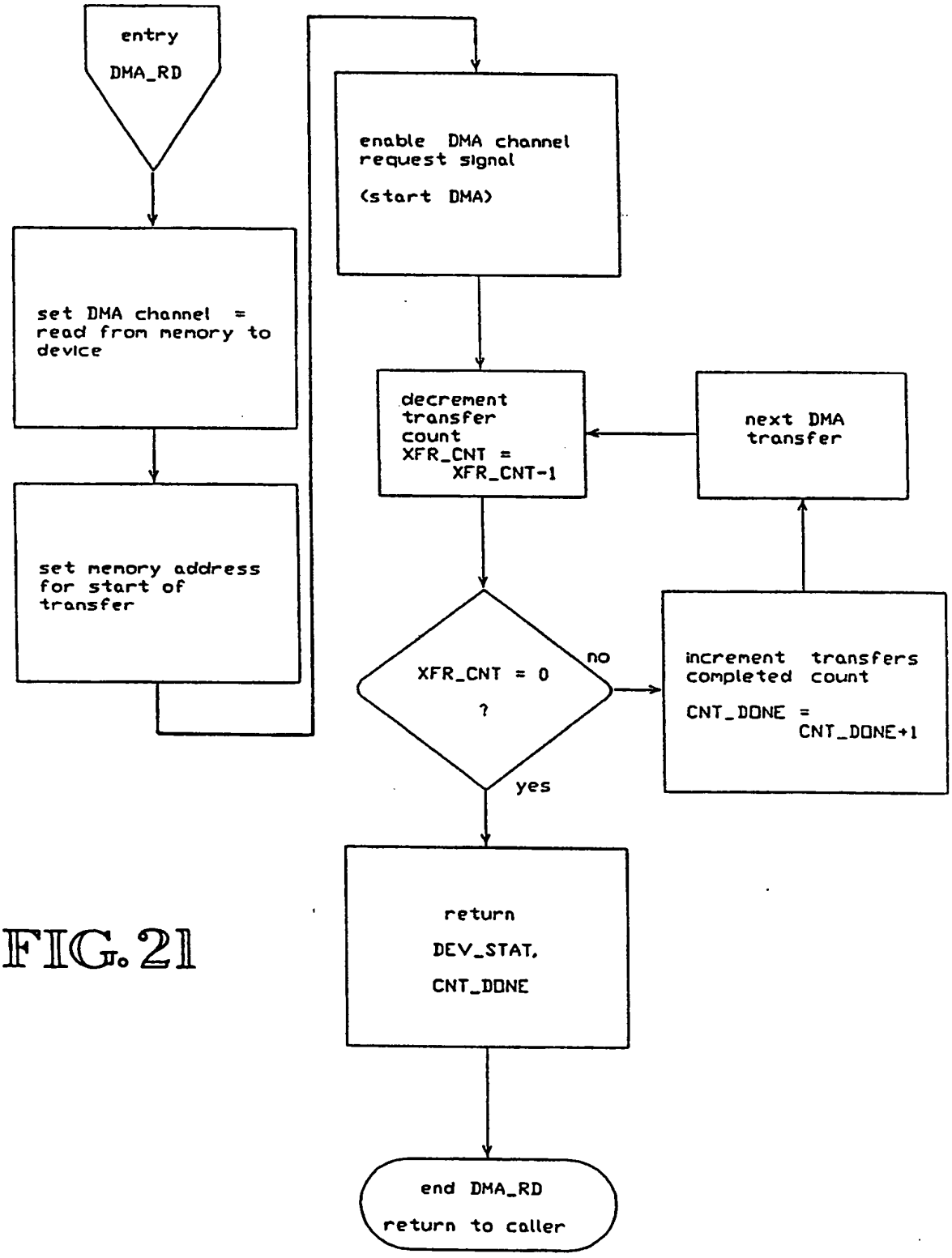

entry
DMA_RD
set DMA channel = read from memory to device
set memory address for start of transfer
enable DMA channel request signal
(start DMA)
decrement transfer count XFR_CNT = XFR_CNT-1
next DMA transfer
XFR_CNT = 0 ?
no
yes
increment transfers completed count
CNT_DONE = CNT_DONE+1
return
DEV_STAT,
CNT_DONE
end DMA_RD
return to caller

FIG. 21

INTERNATIONAL SEARCH REPORT

International Application No PCT/US90/02113

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [3]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC (5) : G06F 12/14
U.S. Cl : 364/200

II. FIELDS SEARCHED

Minimum Documentation Searched [4]

Classification System	Classification Symbols
U.S.	364/200,900 (MSFILE),300; 380/4,25
IPC (5)	G06F 12/14

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [6]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [14]

Category *	Citation of Document, [16] with Indication, where appropriate, of the relevant passages [17]	Relevant to Claim No. [18]
Y	US,A 4,713,753 (BOEBERT ET AL) 15 December 1987 See Figures 1-8, Abstract, & Column 1, Line 10 - et seq.; Column 3, Line 50 - et esq.; Column 5, Line 46 - et seq.; Column 7, Line 7 - et seq.; Column 8, Line 17 - et seq.	1-14
Y	US,A 3,893,084 (KOTOK ET AL) 01 July 1975 See Figures 1-3, Abstract, Column 1, Line 25 - et seq.; Column 3 Line 23 - et seq.; Column 4, Line 5 - et seq.	1-14
A	US,A 4,493,028 (HEATH) 08 January 1985 See Abstract.	1-14
A	US,A 4,652,990 (PAILEN ET AL) 24 March 1987 See Abstract, Column 1, Line 1 - et seq.; Figure 1A,1B,1C.	1-14

(CON'T)

* Special categories of cited documents: [15]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search [2]	Date of Mailing of this International Search Report [2]
04 June 1990	02 AUG 1990
International Searching Authority [1]	Signature of Authorized Officer [20]
ISA/US	Robert B. Harrell for

FURTHER INFORMATION CONTINUED FROM THE SECOND SHEET

A	US,A	4,184,200 (WAGNER ET AL) See Abstract.	15 January 1980	1-14
A	US,A	4,588,991 (ATALLA) See Abstract, Column 1, Line 24 - et seq.	13 May 1986	1-14

V. ☐ OBSERVATIONS WHERE CERTAIN CLAIMS WERE FOUND UNSEARCHABLE [1]

This International search report has not been established in respect of certain claims under Article 17(2) (a) for the following reasons:

1. ☐ Claim numbers . because they relate to subject matter [1] not required to be searched by this Authority, namely:

2. ☐ Claim numbers . because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful International search can be carried out [1], specifically:

3. ☐ Claim numbers_____, because they are dependent claims not drafted in accordance with the second and third sentences of PCT Rule 6.4(a).

VI. ☐ OBSERVATIONS WHERE UNITY OF INVENTION IS LACKING [2]

This International Searching Authority found multiple inventions in this international application as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims of the International application.

2. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims of the International application for which fees were paid, specifically claims:

3. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claim numbers:

4. ☐ As all searchable claims could be searched without effort justifying an additional fee, the International Searching Authority did not invite payment of any additional fee.

Remark on Protest

☐ The additional search fees were accompanied by applicant's protest.

☐ No protest accompanied the payment of additional search fees.

DOCUMENTS CONSIDERED TO BE RELEVANT

A	US, A, 4,797,928 DYKES 10 January 1989 See Abstract, Figures 1-3.	1-14
A	US, A, 4,658,093 HELLMAN 14 April 1987 See Abstract, Figures 1-10, col. 1 (line 4-et seq.), col. 5 (line 39-et seq.).	1-14